UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

A DEEP LEARNING BASED FRAMEWORK FOR PHYSICAL ASSETS' HEALTH PROGNOSTICS UNDER UNCERTAINTY FOR BIG MACHINERY DATA

TESIS PARA OPTAR AL GRADO DE
MAGISTER EN CIENCIAS DE LA INGENIERÍA MENCIÓN MECÁNICA

SERGIO MANUEL IGNACIO COFRÉ MARTEL

PROFESOR GUÍA
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN
VIVIANA MERUANE NARANJO
MARCOS ORCHARD CONCHA

SANTIAGO DE CHILE
2018

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE: MAGISTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: SERGIO COFRÉ MARTEL
FECHA: 2018
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

A DEEP LEARNING BASED FRAMEWORK FOR PHYSICAL ASSETS' HEALTH PROGNOSTICS UNDER UNCERTAINTY FOR BIG MACHINERY DATA

The ongoing development in sensor technology has allowed engineers to monitor complex systems through multisensorial data, generating thousands of data-points in time. This big machinery database is commonly stored to later be used by engineers for reliability purposes through traditional Prognostics and Health Management (PHM) techniques. However, most part of this valuable information is often wasted since PHM methods frequently rely on expert knowledge for their implementation, as well as a good understanding of the physics of failure that govern the system. Hence, to estimate reliability related parameters, such as the State of Health (SOH) or the Remaining Useful Life (RUL) of electrical and mechanical components, data-driven approaches can be applied to complement PHM methods.

In this context, the purpose of this thesis is to develop and implement a novel Deep Learning (DL) framework for the health state estimation of systems and components, based on big machinery data. Accordingly, the following specific objectives are defined: Develop an architecture capable of extracting temporal and spatial characteristics from the data. Propose a health state estimation framework, and validate it using two benchmark datasets: C-MAPSS turbofan engine, and CS2 Lithium-Ion Batteries datasets. Finally, give an estimation of the uncertainty propagation for the health state prognostics yield by the proposed framework.

This thesis proposes a DL framework, which integrates the advantages of spatial management from Convolutional Neural Networks, along with the sequential analysis capabilities from Long-Short Term Memory Recurrent Neural Networks. Dropout is used as a regularization technique, as well as a Bayesian Approximation for the estimation of the uncertainty of the model. Henceforth, the proposed architecture is named CNNBiLSTM.

For the C-MAPSS dataset, four different models are trained, one for each sub-dataset, aimed to estimate the RUL. All four models yield state-of-the-art results for the Root Mean Square Error (RMSE) on their prognostics, showing robustness in the training process and small uncertainty for the test RMSE as well as for the RUL prediction. Similar results are obtained for the CS2 dataset, where the model trained using all battery cells estimates the State of Charge and SOH of the batteries with a lower RMSE than the state-of-the-art results, and a small uncertainty over its estimated values.

Results yielded by the trained models show that the proposed DL framework is adaptable to different systems and can successfully obtain abstract temporal relationship from the sensorial data for reliability assessment. Furthermore, models show robustness during the training process, as well as an accurate output estimation with a small uncertainty.

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE: MAGISTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN MECÁNICA
POR: SERGIO COFRÉ MARTEL
FECHA: 2018
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

MARCO DE TRABAJO BASADO EN APRENDIZAJE PROFUNDO PARA PRONÓSTICO BAJO INCERTIDUMBRE DE SALUD DE ACTIVOS FISICOS PARA DATOS DE MAQUINARIA

El desarrollo en tecnología de mediciones ha permitido el monitoreo continuo de sistemas complejos a través de múltiples sensores, generando así grandes bases de datos. Estos datos normalmente son almacenados para ser posteriormente analizados con técnicas tradicionales de *Prognostics and Health Management* (PHM). Sin embargo, muchas veces, gran parte de esta información es desperdiciada, ya que los métodos tradicionales de PHM requieren de conocimiento experto sobre el sistema para su implementación. Es por esto que, para estimar parámetros relacionados a confiabilidad, los enfoques basados en análisis de datos pueden utilizarse para complementar los métodos de PHM.

El objetivo de esta tesis consiste en desarrollar e implementar un marco de trabajo basado en técnicas de Aprendizaje Profundo para la estimación del estado de salud de sistemas y componentes, utilizando datos multisensoriales de monitoreo. Para esto, se definen los siguientes objetivos específicos: Desarrollar una arquitectura capaz de extraer características temporales y espaciales de los datos. Proponer un marco de trabajo para la estimación del estado de salud, y validarlo utilizando dos conjuntos de datos: C-MAPSS *turbofan engine,* y baterías ion-litio CS2. Finalmente, entregar una estimación de la propagación de la incertidumbre en los pronósticos del estado de salud.

Se propone una estructura que integre las ventajas de relación espacial de las *Convolutional Neural Networks*, junto con el análisis secuencial de las *Long-Short Term Memory Recurrent Neural Networks*. Utilizando *Dropout* tanto para la regularización, como también para una aproximación bayesiana para la estimación de incertidumbre de los modelos. De acuerdo con lo anterior, la arquitectura propuesta recibe el nombre CNNBiLSTM.

Para los datos de C-MAPSS se entrenan cuatro modelos diferentes, uno para cada subconjunto de datos, con el objetivo de estimar la vida remanente útil. Los modelos arrojan resultados superiores al estado del arte en la raíz del error medio cuadrado (RMSE), mostrando robustez en el proceso de entrenamiento, y baja incertidumbre en sus predicciones. Resultados similares se obtienen para el conjunto de datos CS2, donde el modelo entrenado con todas las celdas de batería logra estimar el estado de carga y el estado de salud con un bajo RMSE y una pequeña incertidumbre sobre su estimación de valores.

Los resultados obtenidos por los modelos entrenados muestran que la arquitectura propuesta es adaptable a diferentes sistemas y puede obtener relaciones temporales abstractas de los datos sensoriales para la evaluación de confiabilidad. Además, los modelos muestran robustez durante el proceso de entrenamiento, así como una estimación precisa con baja incertidumbre.

Una tesis es como una cazuela, tiene muchas partes, debe hacerse con paciencia y añadiendo los ingredientes precisos.

En memoria de Víctor Manuel Cofré Moreno, quién dentro de muchas cosas importantes en la vida, me enseñó a disfrutar una buena cazuela.

Acknowledgment

En primer lugar, agradecer como siempre a mi profesor guía, Dr. Enrique López Droguett, quién desde el 2016 ha puesto su confianza en mí para trabajar a su lado y me ha hecho crecer mucho en el área que más me apasiona, la investigación y docencia. Del mismo modo, agradecer a mis profesores Co-guía, Dr. Viviana Meruane y Dr. Marcos Orchard, por haber dedicado parte de su valioso tiempo en hacer ver mis errores, revisar mis trabajos escritos y enseñarme su forma de ver la docencia en la Universidad.

A mis amigos de siempre: Guido, Piero, Rodrigo y Rosario; quienes han hecho que mi trayectoria en la Universidad de Chile sea un disfrutar del conocimiento, lo cual hasta el día de hoy se ve reflejado en largas conversaciones que muchas veces no llegaban a ninguna parte. Son ustedes una de las fuentes de inspiración más fuerte que tengo para dar lo mejor de mí y así seguir mejorando cada día. Los amo.

Me gustaría también dar las gracias a aquellas personas que influenciaron la dirección y resultados de este trabajo de tesis de una u otra forma. Javier, Joaquín, y en general a todos los alumnos de magíster, de los cuales aprendo cada día. En particular, agradecer a Alejandro 'Pelao' Toledo, quien me facilitó su template de tesis y me ahorró mucho trabajo en la escritura, y a María Paz 'Pachi' Valdés por darse el tiempo de revisar la ortografía y gramática de este trabajo. También, me gustaría mencionar a las funcionarias que día a día facilitan nuestras labores, nos alegran por las mañanas con una sonrisa de buenos días, y siempre tienen disponibilidad para favores de último minuto. Soledad, Maricarmen, María Eugenia y Señora Silvia, muchas gracias por todo.

Por último, este trabajo de tesis y eventual título de Magister en Ciencias no hubiese sido remotamente posible de no ser por tres mujeres que son hoy el pilar de mi vida. Margarita Jullian por darme techo, calor, y amor, junto a otra infinidad de cosas sin pedir nunca nada a cambio. Francia Martel, mi amada madre quién ha estado siempre presente para apoyarme y guiarme en lo que sea que quiera realizar, mostrándome mis capacidades incluso cuando yo mismo no creo tenerlas. Finalmente, Camila Correa, a quién simplemente no me dan las palabras para decirte cuánto te amo y de lo feliz que me hace saber cuánto me queda por crecer a tu lado.

Gracias a todos por estar siempre ahí, y espero poder devolverles la mano algún día.

Table of Content

CHAPTER 1 INTRODUCTION ..... - 1 -

1.1 MOTIVATION ..... - 2 -
1.2 OBJECTIVES AND STATEMENT ..... - 4 -
1.2.1 General Objective ..... - 4 -
1.2.2 Specific Objectives ..... - 4 -
1.2.3 Statement and Thesis Scope ..... - 4 -

CHAPTER 2 METHODOLOGY ..... - 5 -

2.1 LITERATURE REVIEW ..... - 5 -
2.2 DEEP LEANING FRAMEWORK PROPOSAL ..... - 5 -
2.3 VALIDATION DATASETS ..... - 5 -
2.4 MODEL TRAINING ..... - 5 -
2.5 RESULTS, METRICS AND UNCERTAINTY ..... - 6 -

CHAPTER 3 BACKGROUND ..... - 7 -

3.1 MACHINE LEARNING ..... - 7 -
3.1.1 Classification and Regression ..... - 8 -
3.1.2 Supervised and Unsupervised Learning ..... - 10 -
3.2 DEEP LEARNING ..... - 11 -
3.2.1 Deep Neural Networks ..... - 12 -
3.2.2 Regularization ..... - 15 -
3.2.3 Convolutional Neural Networks ..... - 16 -
3.2.4 Recurrent Neural Networks and Long-Short Term Memory Cells ..... - 18 -
3.3 DEEP LEARNING IN PROGNOSTICS FOR MECHANICAL COMPONENTS ..... - 20 -
3.3.1 C-MAPSS Dataset for Turbofan Engines RUL Estimation ..... - 20 -
3.4 DEEP LEARNING IN LITHIUM-ION BATTERY STATE ESTIMATION ..... - 25 -
3.4.1 State of Charge ..... - 26 -
3.4.2 State of Health ..... - 26 -
3.4.3 CS2 Dataset for Batteries SOC and SOH Estimation ..... - 27 -
3.5 UNCERTAINTY WITH DROPOUT AS A BAYESIAN APPROXIMATION ..... - 30 -

CHAPTER 4 PROPOSED DEEP LEARNING FRAMEWORK ..... - 31 -

4.1 DATA PREPROCESSING ..... - 31 -
4.2 TRAINING SAMPLES ..... - 32 -
4.3 CONVOLUTIONAL LAYERS ..... - 33 -
4.4 BIDIRECTIONAL LSTM ..... - 34 -
4.5 TRAINING ..... - 35 -
4.6 HYPERPARAMETERS SELECTION AND REGULARIZATION ..... - 36 -

CHAPTER 5 C-MAPSS TURBOFAN ENGINES ..... - 37 -

5.1 MODEL TRAINING, PERFORMANCE AND COMPARISON ..... - 37 -
5.2 UNCERTAINTY MEASUREMENT ON MODELS' ESTIMATION ..... - 39 -

CHAPTER 6 CS2 LITHIUM-ION BATTERIES ..... - 44 -

6.1 MODEL TRAINING, PERFORMANCE AND COMPARISON ..... - 44 -
6.2 UNCERTAINTY MEASUREMENT ON THE MODELS' ESTIMATION ..... - 47 -

CHAPTER 7 CONCLUDING REMARKS ..... - 52 -

7.1 CONCLUSIONS ..... - 52 -
7.2 FUTURE WORK ..... - 55 -

BIBLIOGRAPHY ..... - 56 -

List of Tables

Table 3.1: Non-linear Activation Functions examples. - 15 -
Table 3.2: C-MAPSS Train and Test Sets Resume. - 21 -
Table 3.3: Variables C-MAPSS Dataset. - 21 -
Table 3.4: Sensor Measurements. - 22 -
Table 3.5 : CS2 Lithium-Ion batteries dataset summary. - 27 -
Table 3.6: Variables CS2 Lithium-Ion batteries dataset. - 28 -
Table 4.1: Dropped columns for the training procedure for each dataset. - 32 -
Table 4.2: Time-window length for C-MAPSS dataset. - 33 -
Table 4.4: Hyperparameter values proposed for the architecture’s grid search. - 36 -
Table 5.1: Selected architecture's hyperparameters for the C-MAPSS dataset. - 37 -
Table 5.2: Average results for the C-MAPSS test sets after 10 different trainings. - 38 -
Table 5.3: Uncertainty measures in the C-MAPSS test sets through Dropout. - 39 -
Table 6.1: Selected architecture's hyperparameters for the C-MAPSS dataset. - 44 -
Table 6.2: CNNBiLSTM model training RMSE results for the SOC and SOH. - 45 -
Table 6.3: RMSE [%] SOC comparison when evaluating proposed CNNBiLSTM model with test set. - 46 -
Table 6.4: Uncertainty measures in the CS2 test sets through dropout. - 47 -

List of Figures

Figure 3.1: MNIST digit samples. ..... - 9 -
Figure 3.2: Regression sample for some given data. ..... - 10 -
Figure 3.3: Three-layer Deep Neural Network sample. ..... - 13 -
Figure 3.5: Dropout sample ..... - 16 -
Figure 3.6: Convolution operation. ..... - 17 -
Figure 3.7: Two-layer CNN example. ..... - 18 -
Figure 3.8: Long-Short Term Memory structure example. ..... - 19 -
Figure 3.9: RUL labels for the C-MAPSS dataset. Linear and Non-linear target. ..... - 23 -
Figure 3.10: Voltage discharge curves from Cell 38, for different cycles of operation. ..... - 28 -
Figure 4.1: CNN layers of the proposed CNNBiLSTM framework. ..... - 34 -
Figure 4.2: Bidirectional LSTM layer of the proposed CNNBiLSTM framework. ..... - 35 -
Figure 5.1: RMSE uncertainty for FD001 and FD002 ..... - 40 -
Figure 5.2: RMSE uncertainty for FD003 and FD004 ..... - 40 -
Figure 5.3: RUL estimation under uncertainty. Engine sample from FD001. ..... - 41 -
Figure 5.4: RUL estimation under uncertainty. Engine sample from FD002. ..... - 42 -
Figure 5.5: RUL estimation under uncertainty. Engine sample from FD003. ..... - 42 -
Figure 5.6: RUL estimation under uncertainty. Engine sample from FD004. ..... - 43 -
Figure 6.1: RMSE uncertainty for cells 35 and 36. ..... - 48 -
Figure 6.2: RMSE uncertainty for cells 37 and 38. ..... - 48 -
Figure 6.3: SOC and SOH test RMSE uncertainty for Model 5, trained for all battery cells. .. - 49 -
Figure 6.4: SOC estimation with uncertainty from Model 1. Discharge sample from cell 35.. - 49 -
Figure 6.5: SOC estimation with uncertainty from Model 2. Discharge sample from cell 36.. - 50 -
Figure 6.6: SOC estimation with uncertainty from Model 3. Discharge sample from cell 37.. - 51 -
Figure 6.7: SOC estimation with uncertainty fromModel 4. Discharge sample from cell 38... - 51 -

Abbreviations

ANN	-	Artificial Neural Network.
BiLSTM	-	Bidirectional Long-Short Term Memory.
CALCE-UMD	-	Center of Advanced Cycling Engineering – University of Maryland.
CNN	-	Convolutional Neural Network.
CNNBiLSTM	-	Convolutional Neural Network and Bidirectional Long-Short Term Memory.
DDA	-	Data-Driven Approaches
DL	-	Deep Learning.
ESD	-	Energy Storage Devices.
GPU	-	Graphis Processing Unit
LSTM	-	Long-Short Term Memory.
ML	-	Machine Learning.
MLE	-	Maximum Likelihood Estimator.
MLP	-	Multi-Layer Perceptron.
MSE	-	Mean Square Error.
MTTF	-	Mean Time to Failure
NN	-	Neural Network.
PHM	-	Prognostics and Health Management.
RMSE	-	Root Mean Squared Error.
RNN	-	Recurrent Neural Network.
RUL	-	Remaining Useful Life.
SOC	-	State of Charge.
SOH	-	State of Health.

Chapter 1

Introduction

Ever since the creation of their first tools and machines[1], humans have had to deal with different kinds of degradation and failure problems in their inventions due to corrosion in materials, fatigue in structures, or maybe just poor design choices. These issues became more notorious after the Industrial Revolution, where enormous production plants were created seeking to produce as many goods at the lowest cost as possible, involving hundreds and perhaps thousands of machines that were doomed to unexpectedly fail if proper care was not taken through maintenance. Economies of Scale and production optimization dictate that a fundamental step to effectively achieve such production levels, is to minimize the downtime of the involved equipment. However, predicting the future behavior of a system is not an easy task, which is why Reliability Engineering plays a fundamental role at every stage of the production industry: design, quality control, monitoring, and maintenance planning. For the latter, reliability engineers have usually applied traditional Prognostics and Health Management (PHM) techniques based on statistical approaches to obtain quantitative information from the studied components, where the Mean Time to Failure (MTTF) and the Remaining Useful Life (RUL) are the most known metrics.

PHM techniques have proven to be effective and reliable in their results. Nevertheless, perhaps one of their main handicaps is that each studied component must be treated as a new individual challenge, since not all machines operate in the same manner. For instance, the physics behind the functionality of a cellphone's battery completely differs from the mechanisms observed in the gearbox of a regular automobile. Furthermore, the physics of failure that govern these different phenomena usually lack numerical models for engineers to rely on. Hence, to properly obtain and analyze data from reliability tests for a given system, expert knowledge is required for the broad understanding of its health state. In particular, complex systems that are continuously monitored through sensor measurements, such as temperature, pressure, vibration, amongst others; generate thousands of data-points which contain valuable information related to the correlation of their variables in the temporal space. Most of the time, this information is wasted when analyzed with traditional PHM methods, since they are not sufficient to fully comprehend the future behavior of the system, nor to obtain abstract information from the measured variables.

[1] Considering a machine as any artefact which can execute a given task, such as a bucket to pull out water from a well.

Lately, many limitations presented by traditional PHM methods have been addressed by reliability engineers from a data-driven perspective. Indeed, the ongoing development of Machine Learning (ML) and Deep Learning (DL) techniques, along with computational hardware advances, have allowed engineers to study and implement such algorithms in different areas. For instance, Support Vector Machines (SVM) and Random Forest (RF) are two examples of traditional ML techniques used for classification tasks, which can be applied to classify the health state of a system based on data measured from its monitoring sensors. Moreover, ML approaches such as Support Vector Regression (SVR) and Artificial Neural Networks (ANN) are commonly used for regression tasks, obtaining models which are trained to yield an estimated value of a desired variable based on a given input data. On the other hand, DL techniques aim to tackle similar challenges than ML (classification and regression), but with models highly capable of extracting abstract features from the given data. Architectures such as Convolutional Neural Networks (CNN), Auto-Encoders (AE), Recurrent Neural Networks (RNN), amongst others, have widely been applied to tasks regarding image classification, temporal data analysis, fault detection, and speech recognition, to name a few. These methods are categorized as *deep* learning algorithms since they perform two or more step-analysis over the data, also known as *layers*. However, as powerful as these techniques might be, for many years their application had been limited in areas which require an immediate response, due to the excessive computational resources needed by the algorithms, as is the case for Reliability Engineering problems. Nevertheless, the development of Graphis Processing Units (GPU) enhanced computation, and this has allowed the implementation of DL techniques in the risk and reliability area. Thus, by using data obtained from sensor measurements, PHM methods can be improved or surpassed by data-driven approaches, reducing the computing time needed to yield model results and data preprocessing, and increasing accuracy.

In this thesis, a deep learning framework is presented to estimate the state of health under uncertainty of any system, based on big machinery data. That is, the framework analyses multisensorial data obtained from the equipment monitoring for long periods of time. The approach consists of two convolutional layers to obtain an abstract representation of the sensors' correlation at each time-step, followed by a bidirectional Long-Short Term Memory Recurrent Neural Network for the temporal analysis. Two different datasets are used to train, test and validate the proposed architecture. These consist of a Turbofan Engine and a Lithium-Ion Battery Cell.

1.1 Motivation

For the last decade, data-driven machine health monitoring systems (MHMS) approaches have strongly been introduced in the reliability community, encouraged by the need to accurately predict future behavior of variables that govern complex machinery [1]–[3]. This is particularly true for mechanical components, since these systems are usually provided with online monitoring with sensorial devices such as thermocouples, barometers, and vibration sensors. Machinery data can

be stored to develop indicators which can be used to prevent future loss in production or failure of the system, such as the remaining useful life (RUL) of a component [4], [5].

Given the advantage of automatic feature extraction and the higher level of abstraction that can be achieved from the collected data without specific expert knowledge [6]–[8], deep learning networks have been employed in many reliability problems such as fault diagnostics in gearboxes [9]–[11], sensorial data interpretation [1], [12], [13], and vibration analysis [14]–[17]. For example, Oh et al. [15] proposed a deep learning engineering method for the unsupervised feature extraction from vibration images. The extracted features were then used for classification in diagnostics of a rotational machinery, acquiring a classification accuracy above 95% for three different case studies. Another similar application was used by Shao & Jiang [16], who collected experimental signals from electric locomotive bearings, which were later analyzed with a novel convolutional deep belief network for automatic feature extraction and classification.

Furthermore, as well as in Mechanical Engineering applications, reliability applied MHMS methods have been intensively studied for Electrical Engineering problems. Indeed, the ongoing development in renewable energies and electrical vehicles (EV) [18]–[27] has led researchers to focus their efforts on improving the control and optimization of Energy Storage Devices (ESD) through sophisticated Battery Management Systems (BMS). In this context, it is desired to develop performance indicators such as the State-of-Charge (SOC) and the State-of-Health (SOH), which can allow the user to estimate the autonomy and remaining useful life of these pieces of equipment [28]. The SOC reflects the remaining amount of energy available for usage in the device, according to the maximum energy that it can hold, whereas the SOH gives a measure of the degradation of the battery based on the loss of a given functionality (such as the capacity to store energy).

There are two main approaches for the estimation of such performance indicators of an ESD. On one hand, the most widely studied methods are based on Lumped Thevenin equivalent electric models [29] that characterize the relationship between the battery's SOC and its internal impedance. These models usually consider the energy storage capacity to be unalterable during the discharge process; allowing to approximate the cyclical charge-discharge behavior in laboratory tests at a reasonable rate. On the other hand, the development on Machine Learning and Deep Learning techniques have boosted the use of Data-Driven Approaches (DDA) as a complement to traditional methods for the estimation of the SOC, where Neural Networks are the most commonly used technique for this task [30]–[37]. Nevertheless, as it will further be discussed, there are still many DL applications that are yet to be experimented in this area.

1.2 Objectives and Statement

The objectives and scope for the thesis work are presented as follows.

1.2.1 General Objective

Develop and implement a deep learning-based framework for the health state estimation of physical assets based on big machinery data.

1.2.2 Specific Objectives

- Develop a deep learning architecture capable of extracting temporal and spatial characteristics from big machinery data obtained from multisensorial measurements.
- Create a framework for the health state estimation.
- Apply and validate the framework's performance on the Remaining Useful Life prognostics on the *C-MAPSS* Turbofans dataset.
- Apply and validate the framework's performance on the State of Charge and State of Health prognostics on the *CS2* Lithium-Ion Batteries dataset.
- Give an uncertainty propagation estimation for the health state prognostics yielded by the proposed framework.

1.2.3 Statement and Thesis Scope

This thesis work is intended to propose and validate a deep learning framework to yield accurate prognostics on the state of health of a system. The proposed framework is validated using two different datasets, and the performance results are properly presented in tables and figures.

Chapter 2

Methodology

In order to successfully fulfill the objectives of the present thesis work, the following steps are implemented.

2.1 Literature Review

Until recently, Deep Learning and Reliability Engineering had been treated as completely separated fields. However, nowadays it is not uncommon to see new applications combining both areas published in Journals and Reliability related conferences. Henceforth, it is necessary to implement an extensive literature review of these applications, to identify those areas where an innovative contribution can be made to solve challenges related to the risk and reliability community.

2.2 Deep Leaning Framework Proposal

According to the literature review, a deep learning framework is proposed for the estimation and prognostics of the health state of an equipment. The framework is meant to be applicable to any component summited to cycling stresses, whose operational data was measured and stored for later analysis.

2.3 Validation Datasets

To validate the proposed framework, different datasets are chosen which can later be used to train models. These datasets correspond to the benchmark turbofan C-MAPSS dataset, as well as the Lithium-Ion Battery CS2 dataset from the Center for Advanced Cycle Engineering (CALCE) from the University of Maryland.

2.4 Model Training

For each validation dataset, different models are trained using the proposed deep learning framework. Here, the hyperparameters which define the framework's architecture are tuned to fit

the corresponding datasets. Models are trained ten separate times in order to obtain an estimation on the training robustness of the architecture, and thus evaluate the performance metrics for each model over the average of all training procedures.

2.5 Results, Metrics and Uncertainty

The models' performance is evaluated through different metrics associated with the dataset. For the C-MAPSS dataset, the Root Mean Square Error (RMSE) is used as well as a Score value assigned by the PHM data challenge. CS2 Lithium-Ion batteries dataset is also evaluated with the RMSE metrics. Moreover, for both dataset, an uncertainty measurement of the estimated results is given for the RMSE of each model, along with uncertainty for the estimation value itself.

Chapter 3

Background

The following chapter details the necessary background to successfully achieve the objectives stated for this thesis wok. First, a quick review is given on the definition and purposes of Machine Learning and Deep Learning. In this context, DL techniques are thoroughly explained, addressing their advantages and applications. An exhaustive state-of-the-art literature revision is presented on the applications of DL techniques in health assessment for rotational mechanical equipment, as well as in ESD. Lastly, to validate the proposed DL framework, two datasets are presented along with their respective metrics.

3.1 Machine Learning

Traditionally, to describe a process or phenomena, scientists study the behavior of a system under controlled boundary conditions, seeking to analyze the different effects over the systems' functionality. After many tests and observations, complex specific models can be obtained which are adjusted to describe the acquired data. These models are usually dependent on one or many intrinsic properties of the system. For instance, constants C and m in the Paris' Law (Equation 1) describe how the crack length a changes under a cyclical fatigue stress ΔK over a certain number of cycles N. Another example is the mass m on Newton's famous Second Law (Equation 2), where the force vector $\vec{F}$ needed to accelerate a body by a vector $\vec{a}$, is proportional to its mass. The process of finding these relationships that can accurately model observable phenomena might take months, years or even decades. Sometimes a dead end is hit, wasting valuable time and effort. These models are commonly represented by differential equations or parametric functions and thus can be extremely limited by boundary conditions or properties such as the geometry of the system.

$$\frac{da}{dN} = C\Delta K^m \tag{1}$$

$$\vec{F} = m \cdot \vec{a} \tag{2}$$

In quest of overcoming these issues, Machine Learning techniques became popular in the early 90's. Machine Learning is the field of study that gives computers the ability to learn to execute a specific task without being explicitly programmed for it. Based on a given dataset, ML algorithms use different methods to automatically detect patterns within the data, which are later used to predict future values of determined variables or some other task regarding decision making [38].

In a more formal manner, it is said that a ML computer program (model) learns from experience E (data) to complete a certain task T (output) if its performance on T improves with experience E, where the performance is evaluated by a chosen metric measure P [39]. That is, if the model can *train* itself given new input data and yield better results for the evaluation metric P, then the ML algorithm is capable of *learning* how to model or interpret the desired output variable from the input data. Thus, the performance of any ML algorithm depends heavily on the representation of the data it is given.

Nowadays there are an overwhelming amount of ML techniques designed for thousands and perhaps millions of different applications. However, the main structure of how these algorithms work is common for the parametric models. First, it is said that a parametric ML algorithm is trained to learn from data and perform a given task, meaning that the program has a set of *Parameters* which need to be tuned according to the given training data. Parameters are usually adjusted by minimizing or maximizing a *cost* function, such as the Mean Square Error (MSE), or a Maximum Likelihood Estimator (MLE). The algorithm may also depend on other restrictions set by the programmer beforehand, defined as *Hyperparameters.* To train itself, a ML algorithm uses the training set, where each sample within this data corresponds to a training instance. During the training process, the ML program will become better at one particular task by optimizing the cost function, using only the information in the training set to do so. The performance of the model is evaluated through a metric to be defined by the programmer, which will variate depending on the problem the algorithm is intended to solve (e.g. Root Mean Square Error and Accuracy for regression and classification, respectively), as it will be further discussed in this chapter.

The principal advantage of ML techniques is that they automatically learn to interpret the data without explicitly writing down the general rules that govern the system, nor do they need detailed information about the problem to be solved. Hence, the program is much shorter, easier to maintain, and most likely more accurate. This makes ML algorithms ideal for problems that are either too complex for traditional approaches or have no known models that can describe the studied system. Thus, ML techniques come in handy when dealing with engineering problems where the root cause of a phenomena, such as the mean time to failure of a component, is unknown.

Given that there are several Machine Learning techniques, these are commonly clustered in specific groups depending on the objective they are intended for, and on the information the algorithm is given during the training process. The former separates the algorithms in Regression and Classification tasks, whereas the latter divides them in Supervised and Unsupervised training.

3.1.1 Classification and Regression

One common sorting for ML algorithms comes from the question: *What is the algorithm trying to achieve with the incoming data?* Here, two different approaches arise: Classification and Regression. On one hand, a Classification problem consists on obtaining a model which can map and input data $\boldsymbol{x}$ into a class y_i, with $y_i = \{1, \dots, C\}$, where C is the number of classes. For instance,

one benchmark dataset to test how well a model can perform in a classification task comes from the Modified National Institute of Standards (MINST) dataset[2] [40], where handwritten digits from 0 to 9 are saved in 28x28 pixels images in a grayscale from 0 to 255, such as the ones shown in Figure 3.1. The challenge of this dataset consists on correctly classifying each image to its corresponding digit. As it was previously mentioned, classification algorithms used to solve problems of this nature can be trained either from a supervised or unsupervised approach, depending on the available data, computation power, and the particular interest of the programmer.

Figure 3.1: MNIST digit samples.

A Regression algorithm, on the other hand, is trained to estimate a continuous variable $y = f(\boldsymbol{x})$, where $\boldsymbol{x}$ can be a multidimensional input vector. That is, given a training set $\boldsymbol{x}$, where $\boldsymbol{x_i} \in \mathbb{R}^N$, the regression task consists on finding a mapping function $f(\boldsymbol{x_i})$ that can accurately estimate the value $y_i \in \mathbb{R}$. Figure 3.2 shows two examples of regression models, where the model on the left hand-side uses a linear regression model, which cannot give a proper estimation of the data, whereas the model shown on the right hand-side is a good regression model which could successfully adjust itself to fit the desired output values.

[2] Available at http://yann.lecun.com/exdb/mnist/

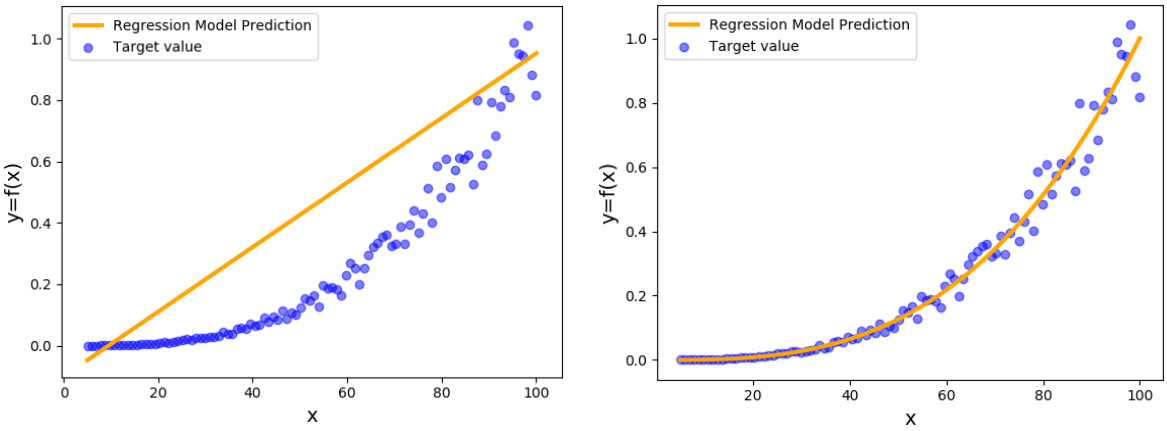

Figure 3.2: Regression sample for some given data.

3.1.2 Supervised and Unsupervised Learning

Another common categorization for ML techniques comes from answering the question: *What information is the program given to be trained?* The answer to this question will depend on whether the training set contains information about the desired values that need to be obtained from the model or not. Supervised models seek to learn a mapping function from an input $\boldsymbol{x}$ to an output y. Where $\boldsymbol{x}$ can be anything from a N dimensional vector, where each dimension represents a feature, as well as more complex data such as images or tensors. Moreover, y can represent either a categorical variable or a real-valued scalar, when dealing with a classification or regression problem, respectively. During the training process, the algorithm is given a training set of labeled data pairs $D_s = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$. That is, for each input data $\boldsymbol{x_i}$ in the dataset D_s, the ML algorithm knows the corresponding label for the output value y_i. Examples of supervised ML techniques are: Support Vector Machines for classification, and Random Forest for regression.

Unsupervised learning models seek to learn new structures within some training dataset $D_u = \{\boldsymbol{x_i}\}_{i=1}^{N}$, where no labels or known outputs are given to train the model. In this case, the tasks consist on an unconditional density estimation looking to build models $p(\boldsymbol{x_i}|\boldsymbol{\theta})$. Since the input vector $\boldsymbol{x}$ will most likely be composed by more than one feature, multivariate probability models need to be created. Unsupervised learning has the advantage that no expert knowledge nor time is needed to label the training data. However proper care must be taken when choosing performance metrics since no knowledge is explicitly given for the training process. Two standard applications of unsupervised training are: Clustering of the training data into classes (clusters) which can later be used to classify new incoming data, and Principal Component Analysis (PCA), used to reduce dimensionality of the data, thus eliminating redundant information within the features and helping to speed data analysis processes.

It is to be noticed that both Supervised and Unsupervised learning models can be used for regression or classification. However, they differ on the data they are given to be trained.

3.2 Deep Learning

Nowadays, ML techniques can be found in several applications, from the speech recognition system used to transform the users' voice into text in a smartphone, to the advertisement offered by webpages to each individual user based on his or her previously searched products. However, processing data in its raw form requires learning capabilities that conventional ML techniques are still limited to perform. This is due to the difficulties regarding the construction of pattern-recognition systems, since they require careful engineering and considerable domain expertise to design a feature extractor which can transform raw data into a suitable internal representation or feature vector for the ML to interpret [41]. That is, it might take more time, knowledge and effort to construct the features to feed a ML algorithm, than training the ML model itself. To overcome this problem, Representation Learning offers a set of methods that allow an algorithm to be fed with raw data and to discover the representations needed for regression or classification by itself. The former is exactly what DL techniques are designed to do.

Deep learning methods consists on ML algorithms with multiple levels of representation, obtained by composing simple non-linear modules that transform the starting raw input into a representation at a higher abstract level. Through the composition of a learning architecture with enough of such simple transformations, very complex functions can be learned from unprocessed data. Hence, DL belongs to one of the many approaches to Artificial Intelligence (AI), which allows computer systems to improve with experience and data [42][3]. DL models can achieve great representative power and flexibility by learning to represent the world as a nested hierarchy of concepts. This hierarchy allows the algorithm to learn complicated concepts by building them out of simpler ones. A graphical representation of this idea would arrange these concepts one on top of each other as a sequence, meaning the graph has a *deepness* associated to it given the many *layers* that constitute its structure. For this reason, this set of machine learning techniques are called deep learning.

Since its first appearance in the mid-20th century, DL has been associated with human reasoning capabilities, and hence they have been thought as an alternative to understand and acquire knowledge from tasks which humans struggle to execute. For instance, AI methods have been used to train chess gaming models through Reinforcement Learning [43], which have been able to defeat the best professional chess players in the world, as well as other parametric models, such as the Stockfish algorithm[4]. Even though the Reinforcement Learning approach acquires state-of-the-art performance in chess, *simpler* programs can still be far superior than human capabilities, since chess is defined by a 64 slots board, with a total of 32 pieces of 6 different kinds which follow a known set of rules. Thus, programming a computer to play chess can be relativity easy.

[3] Deep Learning book available at www.deeplearningbook.org
[4] Stockfish chess engine available at ww.stockfishchess.org

Instead, other challenges rely on the identification of particular features which can only be identified using sophisticated, nearly human-level understanding of the data. Here, traditional ML techniques are not a suitable alternative since it can be very difficult to extract high level abstract features from the raw data. Such challenges are where the true potential of deep learning algorithms can be seen, given that the studied problems do not follow a certain rule or distinguishable pattern that can be directly programmed into an algorithm. One such task is image recognition, where any human can perform accurately at identifying a set of images, however much of the knowledge needed to correctly label or identify an image is subjective and intuitive, and therefore difficult to articulate in a formal way. Nevertheless, DL techniques such as Convolutional Neural Networks (CNN) have proven capable of capturing the knowledge needed to identify and correctly classify images through feature extractors. Indeed, CNN can easily surpass human accuracy, taking less time when dealing with large datasets, and most importantly, behaving in an intelligent way.

Even though the concept of DL algorithms has been around for over five decades, its implementation had been held back due to the high computational power required to tune the great number of parameters within a DL architecture. Furthermore, to accurately train a model, usually big datasets need to be analyzed, resulting on slow training and testing procedures. For the last decade, however, DL algorithms have become popular due to the lower costs on computational components, which has allowed students, researchers, and nearly everyone in the scientific community to have access to fast computers. This, along with the development of GPU powered computation, has boosted the implementation of DL techniques.

3.2.1 Deep Neural Networks

Deep Neural Networks (DNN) are also known as Artificial Neural Networks (ANN) or Multilayer Perceptron (MLP). It is the most basic and common DL structure, consisting on a mathematical function that maps a set of input values to a desired output value. Thus, similarly to other ML algorithms, DNN can create non-linear models for either regression or classification in the form $\boldsymbol{y} = f(\boldsymbol{x}, \theta)$, where θ are the models' parameters. DNN originally receives its name as a reference to the human brain, where multiple neurons are connected to each other and can process data as a flow of information from one group of neurons to another. DNN belong to the "deep networks" branch of ML because their architecture has two or more layers that transform the input data into a more complex representation. To do so, each layer within a DNN performs a linear transformation by multiplying the data from the previous layer by a matrix of weights θ, and adding a bias term b. The output from this linear transformation is then evaluated through a non-linear activation function σ, such as the Rectifier Linear Unit (ReLU) or the hyperbolic tangent (tanh). This idea is formally represented as,

$$\boldsymbol{h_i} = \boldsymbol{\sigma}(\boldsymbol{h_{i-1}^T}\theta_i + b_i) \qquad (3)$$

where $\boldsymbol{h_i}$ is the output from layer i. θ_i and b_i are the layer's weights and biases, respectively, while $\boldsymbol{h_{i-1}}$ is the output from the previous layer $i-1$. The relationship between these layers is the reason why the mapping function created by a DNN is said to be formed by many simpler functions. One way to understand the purpose of each layer is by considering them as successive mathematical functions which provide a new representation of the input. Each layer is further divided into activation units called neurons, where each neuron interacts through Equation 3 with all features from the previous layer.

Figure 3.3 illustrates an example of a DNN architecture, where a three-layer DNN is presented for a regression task. The input layer corresponds to the features from the raw data X, which is connected to one hidden layer to then output a single value. The output given for each hidden unit (neuron) from the Hidden Layer is obtained by Equations 4, 5 and 6. These equations can be resumed in Equation 7, which represents the output of the hidden layer to the output layer, which is identical to Equation 3. It can be seen that each neuron i in the Hidden Layer is assigned one parameter $\theta_{i,j}$ for each incoming feature j from the previous layer (input layer in this particular case). Finally, Equation 8 represents the output layer.

Formally, the output $a_i^{(j)}$ is considered as an activation of neuron i in layer j, where the activation consists on multiplying all input features from the incoming vector from layer $j-1$ by a vector of weights $\Theta_i^{(j)}$. The activation operation from layer j to layer $j+1$ can then be represented by a multiplication by a matrix of weights $\Theta^{(j)}$, which controls the mapping function and with dimensions $s_{j+1} \times (s_j + 1)$, where s_j and s_{j+1} are the number of units in layers j and $j+1$, respectively.

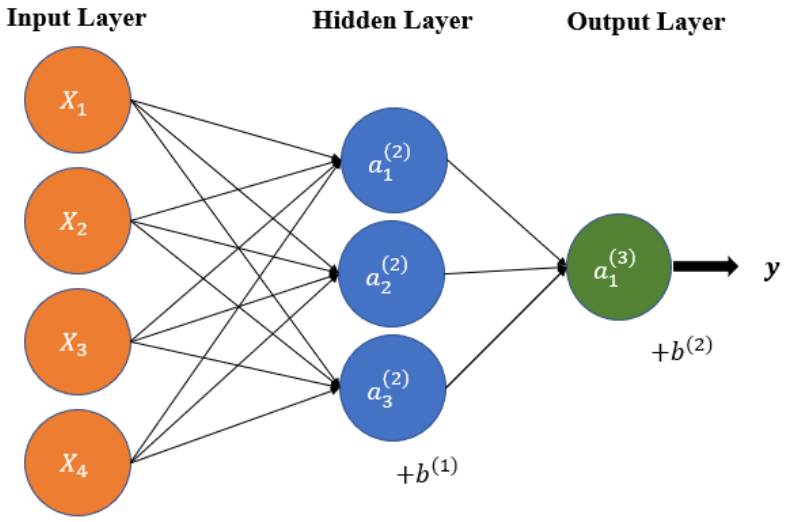

Figure 3.3: Three-layer Deep Neural Network sample.

$$a_1^{(2)} = \sigma\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3 + \Theta_{14}^{(1)}x_4\right) = \sigma(z_1^{(2)}) \quad (4)$$

$$a_2^{(2)} = \sigma\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3 + \Theta_{24}^{(1)}x_4\right) = \sigma(z_2^{(2)}) \quad (5)$$

$$a_3^{(2)} = \sigma\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3 + \Theta_{24}^{(1)}x_4\right) = \sigma(z_3^{(2)}) \quad (6)$$

$$a^{(2)} = \sigma(x^T\Theta^{(1)}) + a_0^{(2)} \quad (7)$$

$$y = a_1^{(3)} = \sigma\left(\Theta_{10}^{(3)}a_0 + \Theta_{11}^{(3)}a_1 + \Theta_{12}^{(3)}a_2 + \Theta_{13}^{(3)}a_3\right) + a_0^{(3)} \quad (8)$$

Activation Functions

The depth concept in deep learning architectures comes from the creation of consecutive non-linear mathematical operations. Particularly, since weights and biases from traditional neural networks are defined to perform a matrix multiplication and an addition, respectively, it is necessary to apply non-linear functions between each layer of a deep neural network. Otherwise, applying a DNN with many layers would be mathematically equivalent to apply a single layer with more neurons in it. These non-linear functions are known as Activation Functions, which can take any non-linear form. Table 3.1 shows a list of commonly used activation functions. However, some functions have become more popular than others in the DL community given their performance during the training and testing procedure for some particular architectures. For instance, the Rectifier Linear Unit (ReLU) is the most applied activation function when training a CNN architecture, whereas the sigmoid and hyperbolic tangent (tanh) are used to create complex structures such as the LSTM networks.

Table 3.1: Non-linear Activation Functions examples.

Binary Step	$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$
sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$
tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$
softplus	$f(x) = \log_e(1 + e^{x})$

3.2.2 Regularization

Due to the usually high degrees of freedom that a deep learning architecture has, it is easy that during the training process overfitting will occur. That is, it is possible to obtain as a result of the training process an over adjustment of the weights and biases of the model to the training data, resulting in poor generalization performance to unseen data (test data). To prevent this, regularization techniques can be applied. One of the most commonly used techniques is dropout [44]. When dropout is used, samples from a Bernoulli distribution r_j^l are used to determine if a feature node j from layer l (e.g. z_j^l, commonly represented by a neuron of the network) is dropped from entering the next layer $l + 1$. This means that when the cost function is being optimized through backpropagation, there are fewer trainable parameters to be tuned per training cycle. This also reduces the dependency of the prediction on a single feature. The value of r_j^l can be 1 or 0 and is drawn from the Bernoulli distribution using Equation 9. The output layer $\tilde{\boldsymbol{z}}^{\boldsymbol{l}}$, after dropout, is obtained using Equation 10. A graphical representation of the dropout technique is shown in Figure 3.4.

$$\boldsymbol{r}_{\boldsymbol{j}}^{\boldsymbol{l}} \sim \textbf{Bernoulli}(\boldsymbol{p}) \tag{9}$$

$$\tilde{\boldsymbol{z}}^{\boldsymbol{l}} = \boldsymbol{r}^{\boldsymbol{l}} \circ \boldsymbol{z}^{\boldsymbol{l}} \tag{10}$$

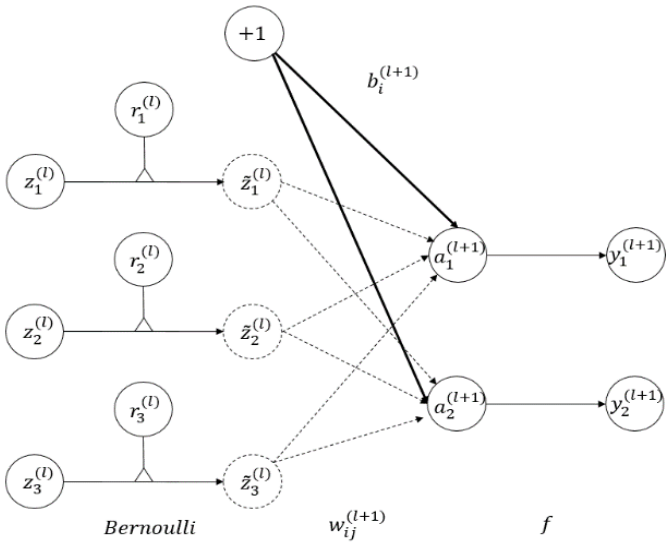

Figure 3.4: Dropout sample

Another regularization technique is early stopping, which stops the training cycle when training and validation errors begin to diverge. These two techniques together greatly reduce overfitting and prevent the network from identifying noise and use it as a distinguishing feature.

3.2.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep learning neural network that uses convolution operations instead of matrix multiplication in its layers. The convolution is performed using a weight matrix ***K***, also known as filter or kernel. The kernel is used to obtain a feature map ***S*** from the input vector ***A,*** using the convolution operation as shown in Equation 11.

$$\boldsymbol{S} = \boldsymbol{A} * \boldsymbol{K} \quad \text{where} \quad \boldsymbol{S}(i,j) = \sum_n \sum_m \boldsymbol{A}(i-m, i-n) \cdot \boldsymbol{K}(m,n) \tag{11}$$

Figure 3.5 shows a representation of the convolution operation using a 2×2 kernel and a 3×3 input data matrix to obtain a 2×2 output matrix. A bias matrix ***B*** is added to the convolution and an activation function is applied to the result to form the feature map ***H*** as shown in Equation 12. The training of the weights and biases can be interpreted as a feature extraction. If a value in the feature map gets activated, it indicates that an important learned feature is in that position. In the case of data analysis, an activation in the feature map can indicate the location of features such as changes in a signal, a frequency or specific shapes. This abstract representation can then be

processed and associated to a physical measurable characteristic, such as degradation, efficiency or production loss, amongst others.

$$\boldsymbol{H} = f(\boldsymbol{A} * \boldsymbol{K} + \boldsymbol{B}) \qquad (12)$$

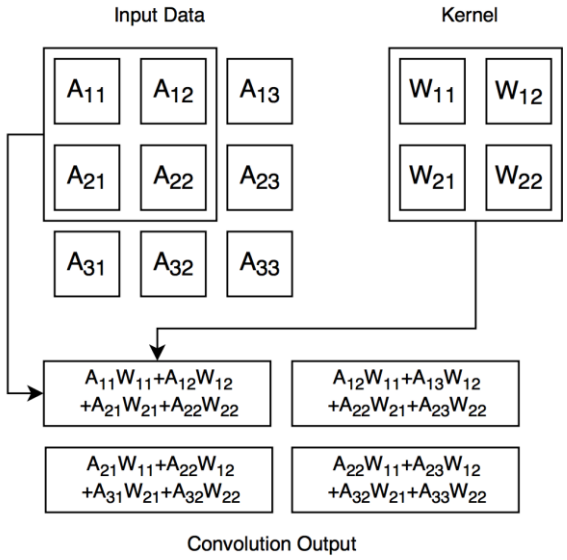

Figure 3.5: Convolution operation.

A convolution layer in a CNN consists of several kernels and biases applied to a single input matrix to generate a set of feature maps in a hidden layer. Every component in the feature map is computed using the same kernel. Also, each component of the output feature map is calculated only from a subset of the input matrix, reducing the amount of connections and, therefore, decreasing the required computation resources. To achieve higher levels of abstraction and more complex relations between features, feature maps can be used as input to other convolution layers.

Usually the last section of a CNN is a feed forward neural network that is responsible for generating the predicted labels as the output vector. Figure 3.6 shows an example of an architecture for a CNN with three 5x5 convolutional filters as the first layer, one 2x2 pooling layer and a fully connected feed forward layer. The CNN is trained in the same way as a NN, defining a cost function to be minimized through an optimizer such as Gradient Descent, Adam or RMSProp.

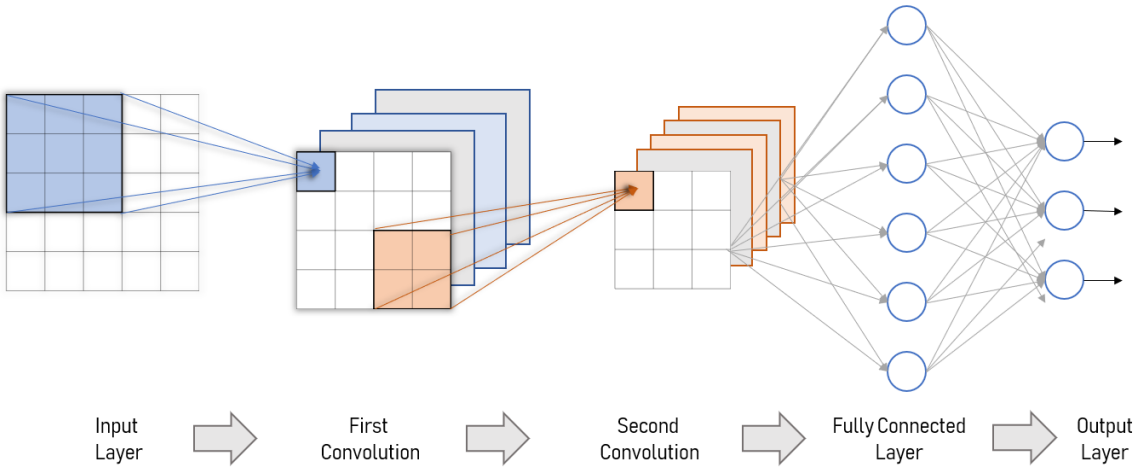

Figure 3.6: Two-layer CNN example.

3.2.4 Recurrent Neural Networks and Long-Short Term Memory Cells

Recurrent Neural Networks (RNN) are one of the most powerful kinds of NN, capable of creating and processing memories of arbitrary sequences of input patterns [45]. However, RNN suffer from optimization problems with long data sequences, making it hard to efficiently tune the networks parameters during the training process and thus causing great computational power demand. To overcome this issue, alternative structures have been proposed, such as the Gated Recurrent Units (GRU) and the Long-Short Term Memory (LSTM) RNN. The classical LSTM structure, called Vanilla LSTM [46], consists of different processes called gates. These gates compute the desired output from a new input data at a time t, along with elements obtained from the previous time-step $t-1$. Equations 13-17 describe these processes, where Equation 17 corresponds to the cell state c^t, which is the main characteristic of the LSTM structure, as it represents a memory capsule containing information of all previous states. The cell state is updated in each time-step through different combinations of Equations 13-15, called input, output and forget gates (Figure 3.7). These gates use a sigmoid activation function to evaluate the linear combination of the new input data x^t with the output of the previous LSTM cell h^{t-1}. Additionally, gate a^t generates the candidates from the inputs x^t and h^{t-1} which will become part of the new cell state when combined with the element wise multiplication of the previous cell state c^{t-1} and the forget gate. Finally, the output h^t is computed by the element-wise multiplication of the output gate with the activation of the calculated cell state (Equation 17).

It is to be noted that gates i^t, o^t, f^t and a^t represent independent NN, which possess their own weights and biases. Hence, after training is complete, each gate will have different outputs. Once the model is trained, it is possible to predict several times steps into the future, given an initial guess value, allowing the study of future behavior of the observed system.

$$i^t = \sigma(W_i\, x^t + U_i\, h^{t-1} + b_i) \quad (13)$$

$$o^t = \sigma(W_o\, x^t + U_o\, h^{t-1} + b_0) \quad (14)$$

$$f^t = \sigma(W_f\, x^t + U_f\, h^{t-1} + b_f) \quad (15)$$

$$a^t = tanh(W_c\, x^t + U_c\, h^{t-1} + b_c) \quad (16)$$

$$c^t = f^t \cdot c^{t-1} + i^t \cdot a^t \quad (17)$$

$$h^t = o^t \cdot tanh(c^t) \quad (18)$$

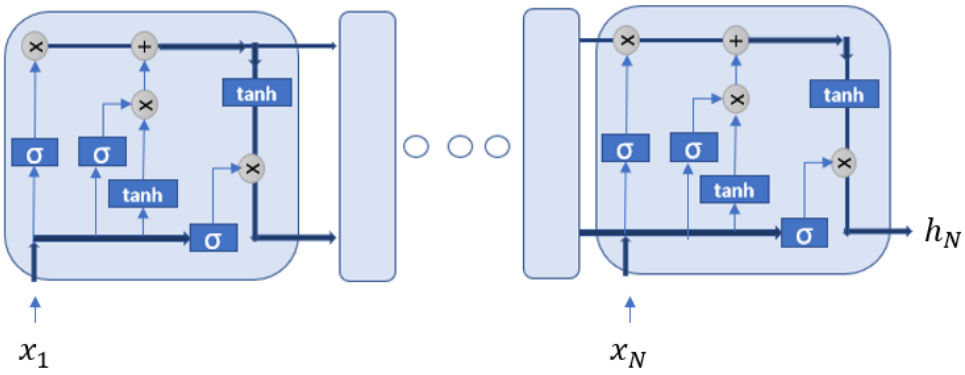

Figure 3.7: Long-Short Term Memory structure example.

3.3 Deep Learning in Prognostics for Mechanical Components

Convolutional Neural Networks are one of many data-driven approaches with a high impact on the reliability field, due to their ability to perform automatic feature extraction from raw data and obtaining a non-linear representation of the signals, thus giving a higher level of abstraction from the data. Most CNN applications are implemented for image classification [47]–[52]. Nevertheless, reliability studies have extrapolated this concept to machinery data analysis [53]–[56]. Indeed, Wen et al. [54] proposed a new CNN architecture based on LeNet-5, creating a 2D image representation of the data, which allowed an automatic feature extraction from the acquired machinery signals. The model was trained for fault classification and tested in a motor bearing dataset, self-priming centrifugal pump dataset, and an axial piston hydraulic pump, achieving an accuracy over 99% for each analyzed dataset. CNN, however, as many other deep learning techniques, were designed for identification and recognition of elements within data arranged as matrices. Given that most sensor measurements are used to monitor changes within different physical variables throughout time, then if the input to the CNN consists only of a single entry of time, it might not contain enough information to correctly predict the future state of a target variable. That is, it is not possible to accurately recognize temporal patterns hidden within the data. Thus, spatial approaches such as NN and CNN are not exactly adequate for the analysis of sequential data.

To address this challenge, different methods have been developed to hold-on to a certain *memory* of the previous state of the system, enabling the prediction of future health state using not only the present information, but also those historical operational conditions of the system that may help in the prediction of the model. One of such techniques are Long-Short Term Memory (LSTM) cells, which are based on Recurrent Neural Networks. This type of model structure has been widely used to achieve state-of-the-art results on sequence modeling tasks such as handwriting recognition [57], automatic rule extraction [58], extreme event forecasting with an end-to-end autoencoder for feature extraction [59], as well as other time series challenges [60]–[62]. Most importantly, deep RNN have successfully been used for machine health monitoring from multi-sensor time series data [63]–[65], obtaining models of the system which can capture the complex temporal behavior of the system, as well as instantaneous dependencies between sensor readings. These dependencies can then be applied to estimate temporal evolutions, e.g., the RUL of a system, such as it was done in [65]–[69]. Hence, in the last couple of years, LSTM have proven to be effective to embrace time series problems related to reliability and maintenance engineering, especially when it comes to prognostics and health management [70].

3.3.1 C-MAPSS Dataset for Turbofan Engines RUL Estimation

In the context of reliability models for mechanical systems, the first dataset used to validate the proposed deep learning framework corresponds to the popular benchmark data generated for the PHM 08' data competition [71]. The dataset corresponds to a multivariate time series obtained

from many simulations from the Commercial Modular Aero Propulsion System Simulation (C-MAPSS), which is divided into training and test sub-datasets. Four different operational conditions are simulated through a determined number of trajectories, which are detailed in Table 3.2. Each dataset is contained in a CSV file, where each row represents one time-step measured in cycles, consisting in 21 sensor measurements, three operational conditions, as well as other useful information as described in Table 3.3. The variables measured by these sensors are detailed in Table 3.4.

Table 3.2: C-MAPSS Train and Test Sets Resume.

Dataset	Train Trajectories	Test Trajectories	Conditions	Fault Mode
FD001	100	100	One (Sea level)	One (HPC Degrad.)
FD002	260	259	Six	One (HPC Degrad.)
FD003	100	100	One (Sea Level)	Two (HPC and Fan Degrad.)
FD004	248	249	Six	Two (HPC and Fan Degrad.)

Table 3.3: Variables C-MAPSS Dataset.

Column	Variable
1)	Unit number
2)	Time in cycles
3)	Operational setting 1
4)	Operational setting 2
5)	Operational setting 3
6)	Sensor measurement 1
7)	Sensor measurement 2
…26)	Sensor measurement 26

Table 3.4: Sensor Measurements.

Sensor Measurement	Measured Variable	Sensor Measurement	Measured Variable
1	Total temperature at fan inlet	12	Ratio of fuel flow to HPC outlet
2	Total temperature at LPC outlet	13	Corrected fan speed
3	Total temperature at HPC outlet	14	Corrected core speed
4	Total temperature at LPT outlet	15	Bypass Ratio
5	Pressure at fan inlet	16	Burner fuel-air ratio
6	Total pressure in bypass-duct	17	Bleed Enthalpy
7	Total pressure at HPC outlet	18	Demanded fan speed
8	Physical fan speed	19	Demanded corrected fan speed
9	Physical core speed	20	HPT coolant bleed
10	Engine pressure ratio	21	LPT coolant bleed
11	Static pressure at HPC outlet		

The data from the C-MAPSS dataset can be considered to come from a fleet of engines of the same type, since each time series is from a different engine (Unit Number). The three operational settings included in the dataset have a substantial effect on the engine's performance. For each trajectory in the training sets, the engine starts operating normally with an unknown initial degradation until failure is detected. The data is also contaminated with sensor noise. In the test set, the time series ends at some time prior to system failure, where the RUL for each test engine is provided in a separated CSV file. However, RUL target is not specified for the engines in the train sets.

Setup and Label Generation

The challenge for this dataset consists on obtaining models through data-driven approaches which can estimate the RUL for each trajectory contained in the test sets. To do so, one model must be trained for each sub-dataset using its respective training data (FD001 to FD004). For each engine (*Unit Number)* in the test set, a RUL label is given for the last measured *Cycle*. However, for the training sets, the RUL label is not provided. Hence, a method must be developed to elaborate the labels for each measured cycle.

There are two supervised approaches that have been implemented to elaborate the labels for the C-MAPSS training sets [53], [65], [72]. The first one considers the literal definition of remaining useful life as the remaining number of cycles to the failure event. Starting from a RUL equal to the length (in cycles) of each training engine, a linear decay represents the degradation of the component, reflected in the RUL label for each cycle of every engine. The second approach is more realistic, where it is assumed that the component degradation cannot be easily detected until a certain threshold R_{early} has been reached. Hence, for the validation of the proposed deep learning framework, the second approach is implemented, with $R_{early} = 125$ cycles as threshold, as it is proposed in [72]. This means that the linear degradation of the engines will start only 125 cycles

before failure. Figure 3.8 shows an example for both approaches for an engine with 300 measured cycles, where the blue line represents an example of the labels used for the deep learning framework validation.

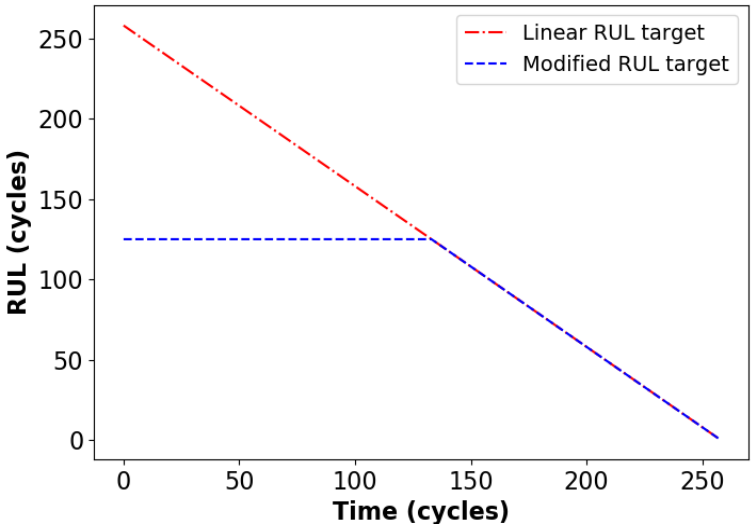

Figure 3.8: RUL labels for the C-MAPSS dataset. Linear and Non-linear target.

Performance Metrics

To evaluate the performance of the trained models for the C-MAPSS dataset, two evaluation metrics were presented for the PHM 08' data challenge: the Root Mean Squared Error (RMSE) and a Score function [65], [71]–[73]. Both metrics are computed based on a parameter h, defined in Equation 12 as the difference between the predicted value by the proposed architecture y_{pred}, and the true label included in the dataset for the test data (y_{true}).

$$h = y_{pred} - y_{true}. \tag{19}$$

RMSE is defined in Equation 20, which is calculated over the N predictions from the corresponding test set. On the other hand, the score function described in Equation 21 outputs different values depending on the sign of h_i, penalizing more those predicted values which are higher than the label (e.g., overestimation of the RUL), since from a reliability point of view this is more dangerous than underestimating the RUL for a data point.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N} h^2} \tag{20}$$

$$Score = \begin{cases} \sum_{i}^{N} (e^{-\frac{h_i}{13}} - 1) & for\ h_i < 0 \\ \sum_{i}^{N} (e^{\frac{h_i}{10}} - 1) & for\ h_i \geq 0 \end{cases} \tag{21}$$

3.4 Deep Learning in Lithium-Ion Battery State Estimation

Accurate State of Charge estimation has been an ongoing research challenge since the massive proliferation of energy storage devices. In this context, Lumped Thevenin equivalent electric models have widely been used to characterize the relationship between the battery's SOC and its internal impedance, considering the energy storage capacity to be fixed during the discharge process. Good results have been achieved through this approach applying techniques such as Unscented Kalman Filtering (UKF) [19], Particle Filtering (PF) [74], Adaptive Cubature Kalman Filter (ACKF) [75], among others [76]–[82]. For instance, Mu et al. [83] proposes the use of an UKF on a Fractional Order Impedance Model (FOIM) – inferred from electrochemical impedance spectroscopy (EIS) – to account for the nonlinearities related to the chemical reactions within a battery.

Furthermore, the development of DL and ML techniques have boosted the use of data-driven approaches as a complement to the aforementioned traditional methods for the estimation of the SOC. From this perspective, NN are the most commonly used techniques for this task [30]–[37]. An example of ML-based estimation frameworks can be found in Yu et al. [84], who proposed a Deep Belief Network (DBN) to update the parameters from the BMS internal SOC model, based on data remotely collected from an EV, achieving SOC estimates with less than a 5% error.

Both data-driven and traditional methods have achieved accurate results on their respective tasks. However, since the performance of a battery is directly affected by its degradation, any SOC estimator that does not consider the SOH of the studied device will be inevitably biased. This is usually reflected in a SOC overestimation for a given energy demand. Thus, different works have been focused on developing models to accurately represent ESD degradation, either by estimating the SOH [85]–[93] or the Remaining Useful Life [94]–[100]. Most of these schemes are based on the maximization of a likelihood function, which in turn relies on the observability of the studied phenomena: there are infinite combinations of battery parameter values that can reflect the same observed voltage and discharge current response. Nonetheless, due to its difficulty, not as many research efforts consider schemes that can jointly estimate the SOC and the SOH [23], [26], [109], [110], [101]–[108].

Hence, it is of interest to characterize both the SOC and SOH of a battery within a small observation time-window, based on the quantification of the impact of discharge profiles in measured variables using a deep learning model. To do so, it is important to clearly define what is considered to be the State of Charge of the system as well as its State of Health, and how these parameters can be characterized from the measured variables of a functional battery.

3.4.1 State of Charge

There are two popular definitions for the state of charge of a battery. The first comes from the widely used *Capacity* concept, which uses the Coulomb Counting Method [36]. Here, to estimate the SOC of a battery the measured current $i(t)$ is integrated in time and normalized by the nominal capacity (C_n) of the battery, i.e., the maximum amount of energy that the battery can hold when it is new. Equation 22 describes this method, where η_i corresponds to the charge (or discharge) efficiency. Now, although this method has successfully been used for SOC estimations, it suffers from a major drawback, which is that it literally counts the amount of current entering (or exiting) the battery. Therefore, it does not meet the energy conservation principle.

$$SOC(t) = SOC(t_0) + 100 \times \int_{t_0}^{t} \frac{\eta_i i(\tau)}{C_n} d\tau \qquad (22)$$

Hence, a better approach is to consider the SOC of a battery cell as the percentage of the remaining energy in the battery E_r, in relation to the total energy charged in the last charging cycle E_{cycle}, as shown in Equation 23.

$$SOC = \frac{E_r}{E_{cycle}} \times 100 \qquad (23)$$

It is to be noticed that E_r takes values in the range $[0, E_{cycle}]$. A quick way to understand the meaning of the definition of the SOC is by thinking of the percentage of battery left in a cellphone. Thus, if the battery is fully charged, the SOC yields a value of 100%, whereas a 0% SOC corresponds to a fully discharged battery. However, the time required to fully discharge the battery is not constant in time, given that the battery degrades each time it goes through a charge-discharge cycle. Therefore, it is imperative to consider the degradation of a battery by estimating its state of health.

3.4.2 State of Health

The State of Health is a representation of the degradation of a battery. However, it is important to understand that there are many factors that take part during the degradation of any component. Thus, the SOH is only an approximation for this degradation, and it may consider one or many parameters for its calculation. One way to represent the SOH of a battery, is by considering the loss of its capacity to hold energy. That is, by comparing the charged energy at each cycle E_{cycle} with

the maximum energy charged in the first charge-discharge cycle E_{max}. Thus, the State of Health of the battery after each cycle can be defined as:

$$SOH = \frac{E_{cycle}}{E_{max}} \times 100 \quad (24)$$

In this case, a 100% and 0% SOH represent a new and completely degraded battery cell, respectively.

3.4.3 CS2 Dataset for Batteries SOC and SOH Estimation

The CS2 Battery dataset from University of Maryland's Center of Advanced Life Cycle Engineering (CALCE) [111], [112] is used to test and validate the proposed model. The dataset corresponds to a multivariate time series obtained from charge-discharge cycles performed at constant current of 1C for four batteries cells until their end of life. These cells underwent the same charging profile which was a standard constant current/constant voltage with a constant current rate of 0.5C until the voltage reached 4.2V. This voltage was then sustained until the charging current dropped to below 0.05A. The discharge cut off voltage for these batteries was 2.7V.

The data for each battery cell is divided into separated files which are labeled according to the date of when the charge-discharge cycles were performed. Each file contains consecutive charge-discharge cycles for a determined date. A summary of the number of files and cycles for each cell is presented in Table 3.5. Furthermore, for each file, every row consists of 17 sensor measurements, which are described in Table 3.6.

Table 3.5 : CS2 Lithium-Ion batteries dataset summary.

Dataset	N Files	N Cycles
CS2 35	26	899
CS2 36	26	927
CS2 37	27	992
CS2 38	27	981

Naturally, battery capacity diminished as lithium-ion cells degraded. As a result, battery discharge tests after 1000 cycles of operation took approximately a quarter of the time associated with a brand-new cell (see Figure 3.9, which shows the voltage measurements of five different discharge cycles for one of the battery cells). Furthermore, as data was acquired using a constant sampling period of 10s in all discharge experiments, each file has a different number of samples.

Table 3.6: Variables CS2 Lithium-Ion batteries dataset.

Column	Variable	Column	Variable
1	Data Point	10	Discharge Capacity (Ah)
2	Test Time (s)	11	Charge Energy (Wh)
3	Date Time	12	Discharge Energy (Wh)
4	Step Time (s)	13	dV/dt (V/s)
5	Step Index	14	Internal Resistance (Ohm)
6	Cycle Index	15	Is FC Data
7	Current (A)	16	AC Impedance (Ohm)
8	Voltage (V)	17	ACI Phase Angle (Deg)
9	Charge Capacity (Ah)		

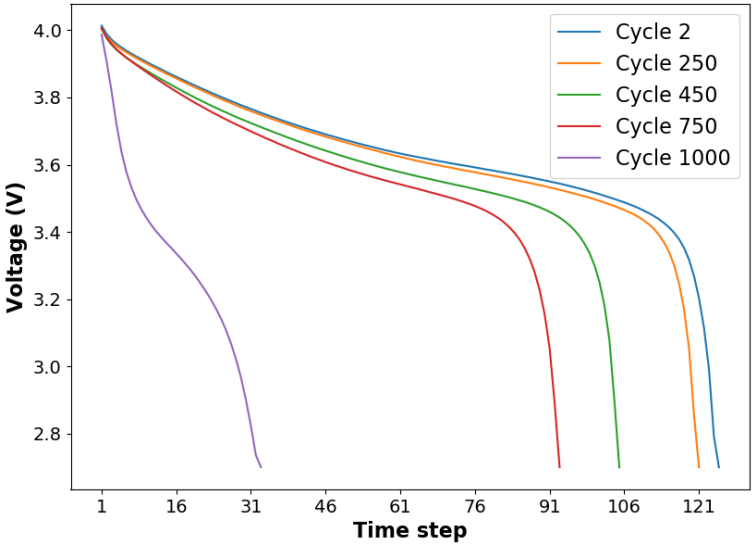

Figure 3.9: Voltage discharge curves from Cell 38, for different cycles of operation.

The CS2 Battery dataset represents a valuable source of information for the characterization of degradation in lithium-ion batteries undergoing cyclic usage profiles at constant ambient temperature. However, it is important to note that the sampling frequency is low (0.1 Hz, compared to 10 Hz in some experiments, where researchers intended to estimate the polarization impedance) and that the discharge current is, by design, constant. The latter fact generates an observability problem in the dynamic time-varying system that represents the evolution in time of battery states; a problem that ultimately does not allow to estimate the internal impedance of the battery (in fact, columns 14, 16 and 17 in the dataset are empty) or to implement joint SOC and SOH estimation approaches based on Thevenin equivalent electric models. Nevertheless, the given data for the dataset is ideal to be tested by the proposed deep learning-based framework, given how the data is presented, which is similar to the C-MAPSS dataset, allowing to test both datasets without modifying the architecture structure.

Setup and Label Generation

Unlike the C-MAPSS dataset, the CS2 lithium-ion batteries do not have a defined task to be used for. Hence, datafiles are not separated in train and test sets, and no labels of any kind are given for the measured instances of the charge-discharge processes. Henceforth, for the validation and performance evaluation of the proposed deep learning framework, this dataset is used to estimate the SOC and SOH of a battery based on a time-window measurement. Using columns 11 and 12 from Table 3.6, the target labels are generated according to Equations 23 and 24. That is, for the SOC labels, the maximum amount of energy discharged for the cycle E_{cycle} is used to as a reference for the remaining energy in the battery cell E_r, whereas for the SOH, E_{cycle} is compared to the maximum amount of energy stored for the first charging cycle E_{max} .

Performance Metrics

The RMSE defined in Equation 20 is used as the main performance metric for the proposed deep learning-based framework validation. The metric is evaluated for the SOC and SOH independently, meaning that for each trained model, two different RMSE are yielded.

3.5 Uncertainty with Dropout as a Bayesian Approximation

Deep learning techniques have successfully been applied for reliability purposes, such as damage assessment [113], [114], vibration signal analysis [55], [70], [115], and RUL estimation for mechanical components [66], [72], [116]–[118]. Nevertheless, these models usually do not give account for the uncertainty of their predictions, which can lead to catastrophic results [119]. Model uncertainty estimation has been an ongoing subject of interest in the reliability community, and a key element for state estimation in the context of Bayesian approaches [120]–[124]. However, until a couple of years ago, deep learning models did not have a reliable estimation of the uncertainty for their predictions. In this regard, Gal et al. [125]–[127] proposed and proved that a deep learning network with non-linearities (activation functions), and with dropout applied before every weighted layer, is mathematically equivalent to a Monte Carlo approximation of a deep Gaussian Process (GP) [128].

Hence, it is possible to use dropout to estimate the uncertainty in fault diagnostic or any other reliability model based on a deep learning network. Given the objective (loss) function of any network with dropout and weight decay regularization, it can be shown that minimizing such function also minimizes the Kullback-Leibler divergence between the approximate and posterior distribution of a deep GP. Hence, it is possible to obtain the uncertainty of the deep network directly from the trained model without any additional assumptions or changes in the network's architecture other than it must be trained considering dropout [44], [129] in every single layer. A mathematical development of dropout uncertainty in LSTM can be found in [130].

A detailed explanation of this approach escapes from the scope of this thesis work. However, it is important to understand that in order to estimate the uncertainty from the test results of a model, several forward passes applying dropout must be performed through the model with the test data, to later analyze the outputs as a Gaussian distribution.

Chapter 4

Proposed Deep Learning Framework

As it was discussed in Chapter 3, many deep learning techniques have been applied for sensor interpretation and analysis of multisensorial measurements. Convolutional and Recurrent Neural Networks have shown great performance on pattern recognition and temporal analysis, respectively. However, only a small number of efforts have been presented combining CNN with LSTM techniques. On one hand, Chen et al. [13] proposed a deep CNN combined with a deep bidirectional LSTM (BiLSTM) network for a classification task on building occupancy estimation, analyzing data from environmental sensors. On the other hand, Zhao et al. [64] also trained a deep network combining CNN and BiLSTM for a regression task on tool wear prediction. Although both approaches show accurate results on their respective tasks, convolution layers are used to compute temporal relationships from the data, which implies a double computation in the temporal space since this is the task intended for the BiLSTM.

Thus, the proposed health state estimator framework is based on a hybrid CNN and LSTM architecture. The framework is designed to take advantage of the CNN's capability in building higher abstraction representations for spatial recognition and the LSTM's ability to deal with sequential analysis. A non-linear RUL target is proposed as an objective. Before training the resulting model, raw data is normalized with a Min-Max scaler in the range [0,1], and those variables from the dataset that do not contribute valuable information to the model, or were used for the label generation, are dropped. A cross-fold validation is performed to select critical hyperparameters. Once these hyperparameters are selected, the proposed architecture is trained and tested through the validation datasets described in Chapter 3.

4.1 Data Preprocessing

For each dataset, a Min-Max normalization is used along every variable to normalize the data and thus to avoid overfitting of the model over variables with a high order of magnitude. Given a vector X, its scaled value X_{scaled} is calculated as described in Equation 25. Where X_{min} and X_{max} correspond to the minimum and maximum value of each column feature, respectively.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}. \qquad (25)$$

Furthermore, variables which do not change in time, and hence could bias the results, are dropped. A similar approach is taken to select variables for the CS2 dataset. Here, same as the C-MAPSS dataset, those variables from Table 3.6 which do not change in time, are dropped. Also, variables that are not relevant for the charge-discharge process or were used to generate the labels for the SOC or SOH, are not used during the training process. Hence, from Table 3.3 and Table 3.6, the columns dropped for the C-MAPSS and CS2 datasets, respectively, are listed in Table 4.1.

This results in a total of 14 sensor measurements to train model for the C-MAPSS dataset, and 5 sensor measurements for the CS2 models: Current, Voltage, Charge capacity, Discharge Capacity, and dV/dt.

Table 4.1: Dropped columns for the training procedure for each dataset.

C-MAPSS Turbofan Engines	**CS2 Lithium-Ion Battery Cells**
Unit Number	Data Point
Time in Cycles	Test Time
Operational Settings 1	Date Time
Operational Settings 2	Step Time
Operational Settings 3	Step Index
Sensor measurement 1	Cycle Index
Sensor measurement 5	Charge Energy
Sensor measurement 6	Discharge Energy
Sensor measurement 10	Internal Resistance
Sensor measurement 16	IS FC Data
Sensor measurement 18	ACI Impedance
Sensor measurement 19	ACI Phase Angle

4.2 Training Samples

Unlike regular NN, Convolutional Neural Networks apply a convolution operation instead of a matrix multiplication. This operation was originally proposed to recognize shapes and patterns in images through the mathematical relationship between nearby pixels. However, it is possible to extrapolate this concept to monitor and analyze how the correlation among sensor measurements change throughout time while the measured equipment deteriorates. To do so, a sliding-window of length N_t is used to transform the N_{ft} sensor measurements or features into multiple samples from the dataset. Therefore, each sample is arranged in a 2D matrix of shape $N_t \times N_{ft}$, which contains all sensor measurements for a given period of time. These matrices are then used as input for the first two layers of the proposed architecture, which correspond to convolutional layers.

For the C-MAPSS dataset, the number of selected sensor measurements after the preprocess of the data is $N_{ft} = 14$. As for the sample time length N_t, it must be considered that the cell state (Equation 17) allows the LSTM to handle longer sequences than other techniques such as the RNN. Thus, the more temporal information the LSTM is given, the better it performs. Yet, if the chosen time length is too large, then there will be too many parameters to be tuned, slowing both the training and testing processes.

The work put forward by Li [56], proposed that for a deep CNN (DCNN), the optimal sample length is $N_t = 30$ cycles when comparing accuracy and training time, considering that the shortest test sequence has 31 time-steps. However, the shortest sequences for the test engines from the FD002 and FD004 test set correspond to 21 cycles and 19 cycles, respectively. Thus, it is not possible to use a longer time-window, and therefore each sub-dataset from the C-MAPSS dataset is analyzed with a time-window length equal to the smallest test engine sample. Table 4.2 shows the adopted value for each dataset.

Table 4.2: Time-window length for C-MAPSS dataset.

Dataset	N_T
FD001	30
FD002	21
FD003	30
FD004	19

On the other hand, for the CS2 dataset, the number of selected features is $N_{ft} = 5$. However, since there are no defined train and test sets for this dataset, for the sample time length, choosing the time-window length is less intuitive. Here, it must be considered that as battery cells deteriorate, the discharge cycle shortens in time. As illustrated in Figure 3.9, over 75% of the total cycles have discharge length is over three times longer than the last cycle discharge cycle. Hence, a sample length equal to the shortest discharge cycle of the dataset is a suitable measure, which for the CS2 dataset corresponds to $N_t = 24$.

4.3 Convolutional Layers

The first two layers of the proposed framework's architecture consist on two CNN layers, as illustrated in Figure 4.1. The first convolution operates a filter of size 1×2 with a stride $s = [1,2]$. That is, the convolution takes only sensorial data measured at the same time-step, seeking to obtain an abstraction representing the relationship between every couple of sensors. The filter is applied FM_1 times, thus generating FM_1 feature maps of size $N_t \times N_{ft}/2$ after the first convolution, which are then evaluated with an element-wise ReLU activation function. It can be noticed that each

feature map has the same length as the original sample 2D matrices, however each feature (column) is a new representation from the sensor measurements for each time-step.

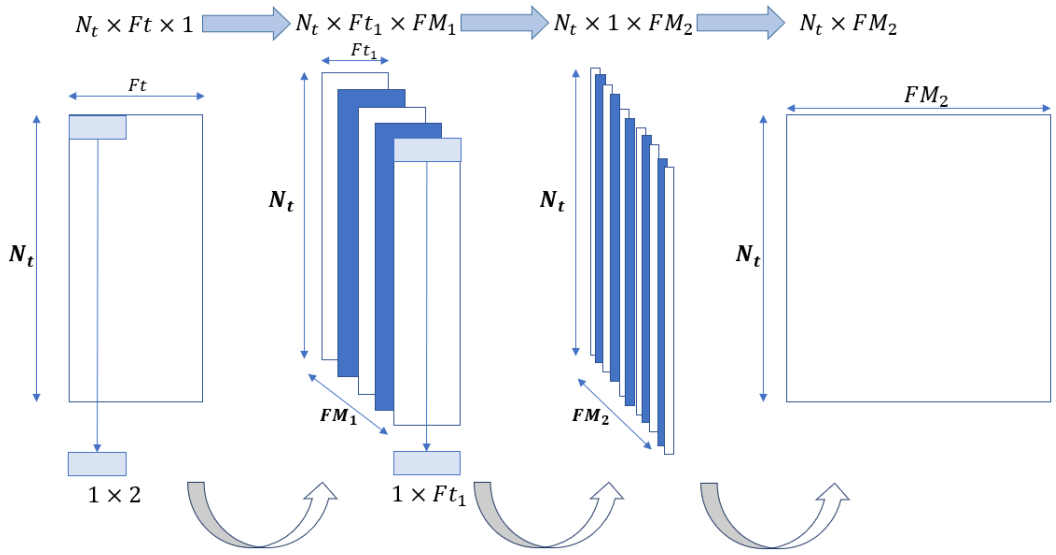

Figure 4.1: CNN layers of the proposed CNNBiLSTM framework.

Once the first convolution has been performed, the created feature maps are used as inputs into the second convolutional layer, which this time operates with a filter of size $1 \times N_{ft}/2$. Analogous to the first convolution, the second layer is applied only to features from the same time-step, and then evaluated with an element-wise ReLU activation function. However, on this occasion, the convolution operation seeks to get a relationship between all features for each time-step. The convolution filter is applied FM_2 times, thus generating FM_2 feature maps of size $N_t \times 1$. These feature maps are then reshaped into a new matrix, where each column corresponds to one feature map from the second convolution layer. Thus, the output 2D matrix has a shape of $N_t \times FM_2$, which is then used to feed the third layer corresponding to the BiLSTM.

4.4 Bidirectional LSTM

The third and last layer of the proposed architecture consists of a BiLSTM comprising two independent LSTM cells that run in parallel, as shown in Figure 4.2. One of the cells processes the input data in sequential order, that is, starting from the first time-step to the last data point N_t. The second LSTM cell takes the sequential data in the reverse order, starting from time-step N_t and ending in the first sequential entry. These cells are called forward and backward cells, respectively. In the proposed architecture, both forward and backward cells take the higher abstraction representation generated after the first two convolutional layers as their input. Each LSTM outputs the hidden units of the cell gates (Equation 17). The outputs of the cells are then concatenated and

used as an input to a fully connected layer, yielding the predicted variable(s) y_{pred}, corresponding to the RUL and SOC & SOH for the C-MAPSS and CS2 datasets, respectively.

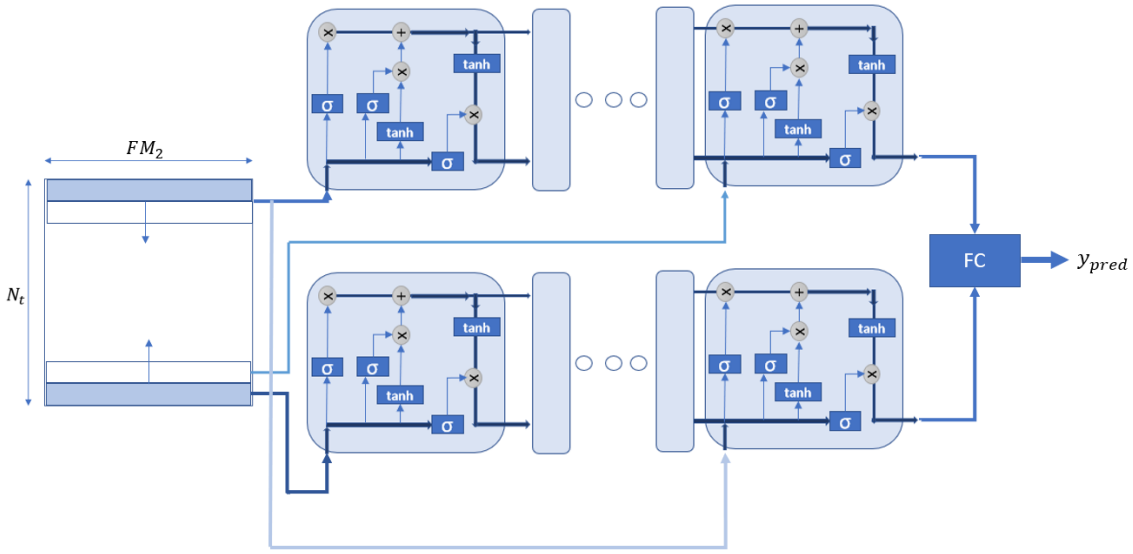

Figure 4.2: Bidirectional LSTM layer of the proposed CNNBiLSTM framework.

4.5 Training

For the training procedure, the created input 2D matrices need to be divided in train and test sets. Then, the train set is again divided into a training and a validation set. In the case of the C-MAPSS dataset, the train and test sets are already separated, and hence the training and validation sets are randomly split in 80% and 20% of the train set, respectively. However, the CS2 dataset does not have a defined train and test set. Therefore, the created input 2D matrices are randomly divided in three sets: Training set (60%), validation set (20%), and test set (20%). The model is trained over the training set, leaving the validation set to evaluate the performance of the model at each training epoch. Samples from the training set are fed in batches to the model. The model is optimized through backpropagation with the RMSProp optimizer [85]. Training is performed in an Intel Core i7 6700K CPU, 32 GB DDR4 (2400 MHz) RAM and a NVIDIA Titan XP GPU of 12 GB memory. For computation, Python 3.5 language with Tensorflow 1.4 [86] was used, along with cuDNN 5.1 and Cuda 8.0 libraries. Ubuntu 64 bits 16.04 LTS was used as operating system.

4.6 Hyperparameters Selection and Regularization

The proposed architecture has several hyperparameters that need to be selected carefully. These hyperparameters are: the number of feature maps for each convolutional layer, the number of hidden units in the LSTM cells, the batch size fed to the network during training and the number of epochs when training. To select the best model, a grid search is performed over every combination of the hyperparameter values presented in Table 4.3, which are trained to fit each of the datasets presented in this Chapter, giving a total of 3072 different models to be trained.

To prevent overfitting, two regularization methods are implemented. The first method corresponds to dropout [81], which is applied at each convolutional layer, as well as both cells from the BiLSTM and the following fully connected layers which yield the final output of the network. The second regularization method is early stopping, where the training process is stopped if the model's accuracy does not improve over the validation set but does for the training set for three consecutive epochs.

Table 4.3: Hyperparameter values proposed for the architecture's grid search.

Feature Maps 1st Convolution	**Feature Maps 2nd Convolution**	**Epochs**	**Batch Size**	**Number Hidden Units LSTM**	**Dropout Probability**
16	32	50	64	50	0.5
32	64	75	128	100	0.6
64	128	100	256	150	0.7
128	256	125	512	200	-

Chapter 5

C-MAPSS Turbofan Engines

A grid search is performed for each sub-dataset in the C-MAPSS dataset, to select the hyperparameters of the proposed model, according to Chapter 4. This is done considering all the hyperparameters listed in Table 4.3. Out of the 3072 possible combinations, the selected hyperparameters which yield the best average results for all four sub-datasets are reported in Table 5.1. In addition to the performance over the datasets, the aforementioned procedure showed that both the batch size used for training as well as the number of hidden units in the LSTM, have the greatest influence on the training time of the network. For instance, training for 200 LSTM hidden units takes about four times more than for 50 LSTM hidden units, keeping all other hyperparameters constant. A similar effect is caused by the batch size, which is more intuitive since the bigger the batch size, the fewer iterations per epoch are needed to train a model for the same number of epochs.

Table 5.1: Selected architecture's hyperparameters for the C-MAPSS dataset.

Feature Maps 1st Convolution	Feature Maps 2nd Convolution	Epochs	Batch Size	Number Hidden Units LSTM	Dropout Probability
32	128	75	256	50	0.4

5.1 Model Training, Performance and Comparison

With the selected hyperparameters, ten different models are trained and tested for each sub-dataset, applying a 40% dropout probability after each layer of the proposed architecture. Table 5.2 shows the average and standard deviation obtained for the evaluation metrics described in Chapter 3 for each sub-dataset. The results are compared with other relevant research works done over the same dataset. Indeed, Zheng et al. [65] trained a deep LSTM network, while Zhang et al. [131] proposed a Multi-objective Deep Belief Network Estimator (MODBNE) for the RUL estimation of the C-MAPSS dataset. However, both articles report results only for one model trained for each sub-dataset, thus making the results difficult to compare.

So far, the best average results obtained for the C-MAPSS dataset were reported by Li et al. [72] who trained a Deep Convolutional Neural Network (DCNN) with four convolutional layers of ten

feature maps each. From Table 5.2, it can be seen that the proposed CNNBiLSTM-based model outputs a better average RMSE than any of the other state of the art models [65], [72], [131], giving also a similar score value, except for the FD003 sub-dataset where the proposed model does not perform as well as Li's model [72]. The latter can be associated to the fact that the RUL target from the test sets are not modified to meet the R_{early} criteria stated Chapter 3. In particular, the trained model will not be able to correctly predict 15% of the FD003 test data. Instead, a more conservative underestimation will be yielded for the RUL in these cases. A similar effect is presented for the rest of the trained models, implying that even though the test set has not been modified in any way, better performance is obtained for each test set with score values within the same order of magnitude. For the RMSE metric, the proposed CNNBiLSTM base model gives a 20% improvement over FD001, a 6% over FD002, a 11% FD003, and a 3% over FD004 sub-dataset.

Table 5.2: Average results for the C-MAPSS test sets after 10 different trainings.

		MODBNE [125]		**DCNN [73]**		**DLSTM [65]**		**CNNBiLSTM**	
Dataset	Metric	Mean	STD	Mean	STD	Mean	STD	Mean	STD
FD001	RMSE	15.04	-	12.61	0.19	16.14	-	10.09	0.60
	Score	334	-	273	24	338	-	339	72
FD002	RMSE	25.05	-	22.36	0.32	24.49	-	21.02	0.31
	Score	5585	-	10412	544	4450	-	10830	1788
FD003	RMSE	12.51	-	12.64	0.14	16.18	-	11.34	0.70
	Score	421	-	284	26	852	-	1189	167
FD004	RMSE	28.66	-	23.31	0.39	28.17	-	22.74	0.42
	Score	6557	-	12466	853	5550	-	9849	954

From the presented results in Table 5.2 and the dataset description in Table 3.2, it can be observed that the average RMSE for the FD002 and FD004 datasets are about two times bigger than for the FD001 and FD003. Furthermore, the average Score values from the former-datasets surpass the later by more than one order of magnitude. Hence, the proposed CNNBiLSTM-based model struggles more to yield an accurate prediction when dealing with six operational conditions instead of one, rather than adding a second failure mode to the setup.

Besides its performance, the proposed architecture also presents other advantages over the abovementioned models. For example, the trained models take in average 75 epochs to converge, while when training a DCNN can take over 250 epochs and a DLSTM needs over 2000 epochs. Furthermore, the CNNBiLSTM-based model has an average training time for the selected hyperparameters of 64.3s, with an average test evaluation time of 0.01s for a new input data, thus being suitable for real time monitoring.

5.2 Uncertainty Measurement on Models' Estimation

The proposed CNNBiLSTM-based model accounts for the model uncertainty on the RUL estimation. This is accomplished by means of two different approaches. First, 5,000 forward passes are performed on a trained model for the test set of each sub-dataset, setting a dropout probability of 10%, thus obtaining different RMSE values at each iteration. These values can then be used to determine the uncertainty of the model on the test RMSE by taking the average and standard deviation of the test RMSE obtained at each iteration, as it was mentioned in Chapter 3.

Table 5.3 summarizes the results for this approach, as well as the results for the Shapiro-Wilk null hypothesis test [132], which is used to evaluate the normality of the yielded distributions. The null hypothesis states that the data was given by a Gaussian distribution, where this hypothesis is rejected for a low test-statistic (*W*) value, or if the *p-value* is lower than the desired significance level. In Table 5.3, it can be seen that all *W* values obtained from the null hypothesis yield the highest possible value (good fit) with all *p-values* higher than 10% significance. Thus, the null hypothesis can be accepted, meaning that a Gaussian distribution can represent the RMSE uncertainty for the trained models. It must be noticed that these Gaussian distributions are a direct result from applying dropout while testing, and it was not imposed in any way to the models during the training procedure.

Table 5.3: Uncertainty measures in the C-MAPSS test sets through Dropout.

	RMSE			
Sub-dataset	**Mean**	**STD**	**W**	**p-value**
FD001	10.09	0.31	1,00	0,66
FD002	21.40	0,21	1,00	0,63
FD003	11.74	0,33	1,00	0,12
FD004	22,98	0,21	1,00	0,15

Figure 5.1 shows the normalized probability density function for the FD001 and FD002 test RMSE, showing significant robustness given that the results output a normal distribution with a small standard deviation. That is, the uncertainty over the test RMSE results (Table 5.2 and Table 5.3) is small.

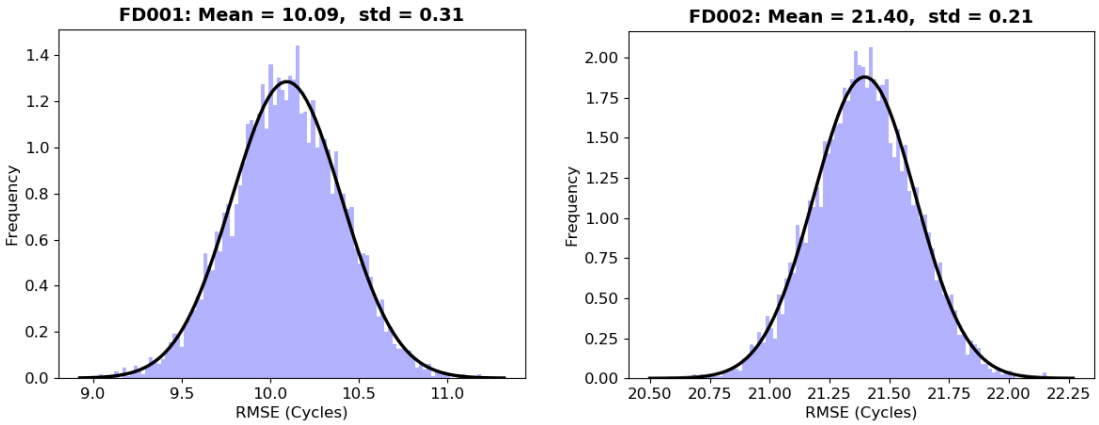

Figure 5.1: RMSE uncertainty for FD001 and FD002

Similar results are obtained when evaluating the test RMSE's uncertainty for models trained for the FD003 and FD004 sub-datasets. The latter is shown in Figure 5.2, where once again a small standard deviation reflects a small uncertainty over the models' error. It is to be noticed that Figure 5.1 and Figure 5.2 yield a mean RMSE similar to the ones reported in Table 5.2, evidencing the robustness on the estimation precision of the proposed CNNBiLSTM model.

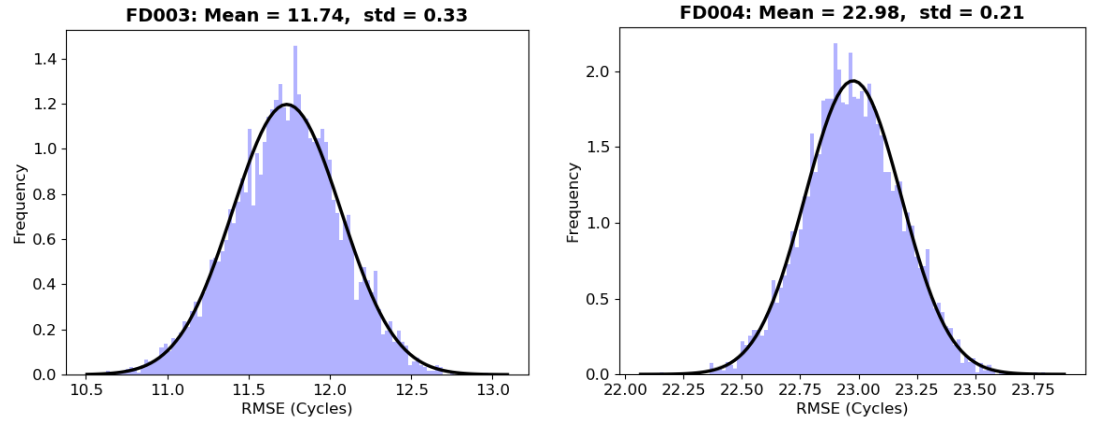

Figure 5.2: RMSE uncertainty for FD003 and FD004

The second approach to estimate the models' uncertainty, consists in taking an entire run of one sample engine from the training set of each sub-dataset. 2D matrices are generated for every time-step of the run, as it was described in Chapter 4. For each matrix, 10,000 forward passes through the trained model are performed with a 10% dropout probability, enabling the estimation of uncertainty for the predicted RUL at each cycle of the engine's sample run.

The aforementioned procedure is performed for one random sample engine for each sub-dataset, evaluating the created 2D matrices over their corresponding model. Figure 5.3 shows the predicted RUL with a 90% probability interval for each cycle in the sample engine for the FD001 sub-dataset.

It can be observed that the predicted RUL adjusts smoothly to the target RUL before R_{early} is reached. Afterwards, the model tends to underestimate the RUL value at the beginning of the target RUL, which is conservative from a reliability point of view. Later, the prediction converges to the true RUL value for the last cycles of the run with a smaller uncertainty than before.

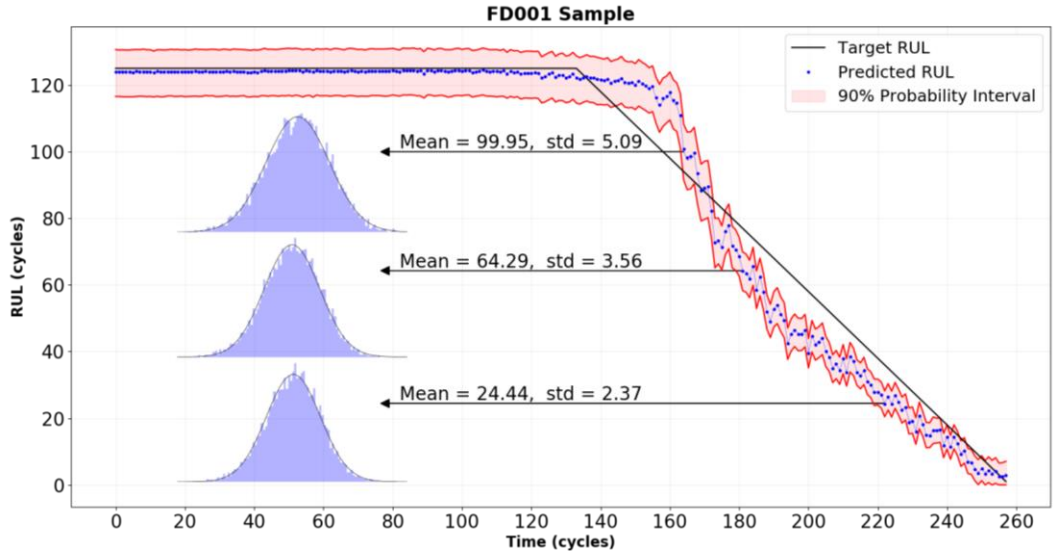

Figure 5.3: RUL estimation under uncertainty. Engine sample from FD001.

A similar behavior can be observed when testing different engines from sub-datasets FD002, FD003 and FD004, as illustrated in Figure 5.4, Figure 5.5 and Figure 5.6, respectively. In this case, unlike the other three models, the FD002 model tends to overestimate the RUL after R_{early} has been reached. However, it converges to the expected values when moving towards the failure event. It must be mentioned that even though these are only one single example of each dataset, it can be seen from the probability interval showed along the predicted values that the uncertainty decreases towards the failure event. This is corroborated by the three histograms taken from different points throughout the engine run, where the standard deviation of the estimated RUL decreases over time for all the test engine runs.

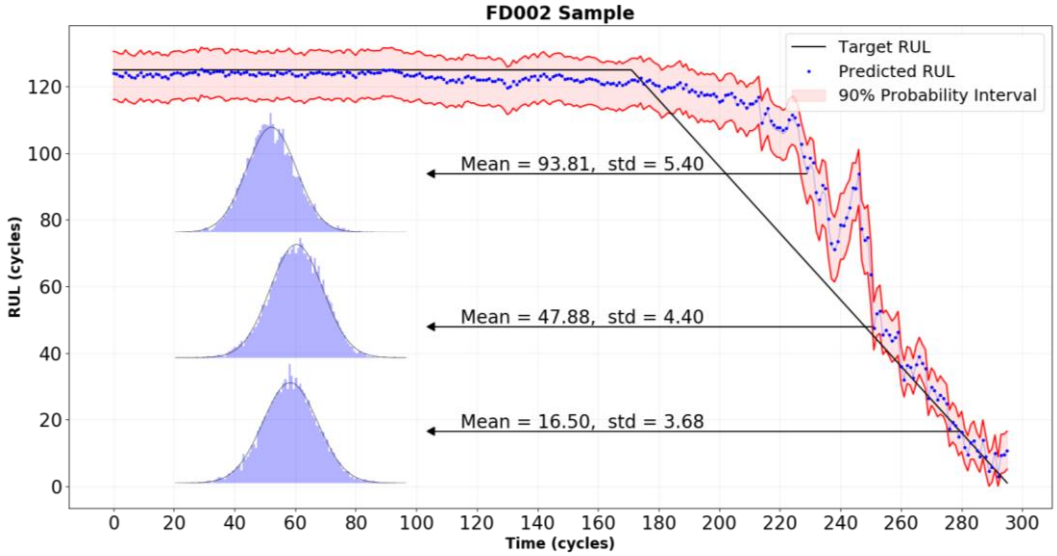

Figure 5.4: RUL estimation under uncertainty. Engine sample from FD002.

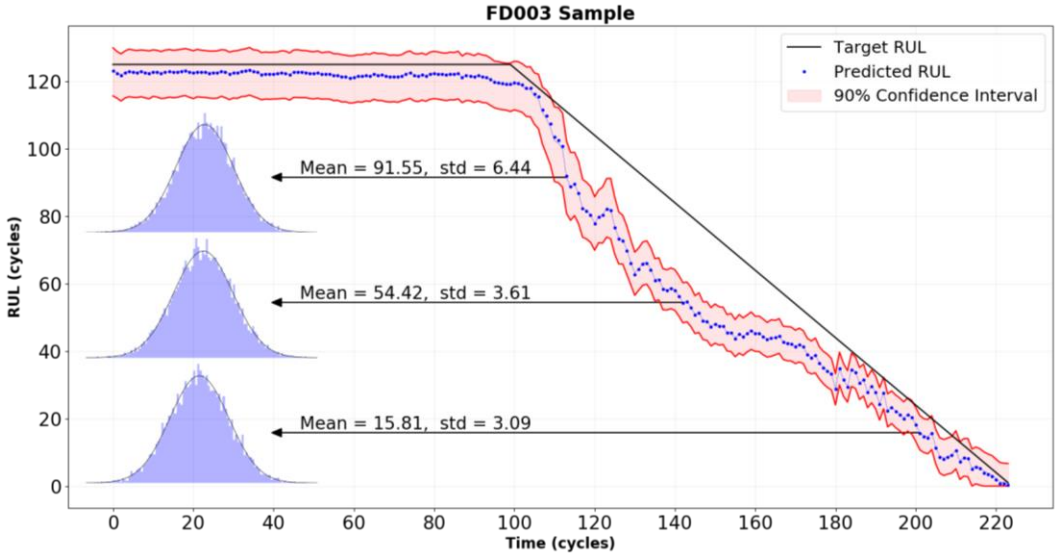

Figure 5.5: RUL estimation under uncertainty. Engine sample from FD003.

The discussed results show that the trained models can successfully interpret the behavior of the RUL curve throughout the run. This is a remarkable conclusion for the proposed CNNBiLSTM framework since the studied dataset had different operational conditions and failure modes. Particularly, this is true for the results yielded by the model trained for the sub-dataset FD004, which has the most challenging conditions and two failure modes.

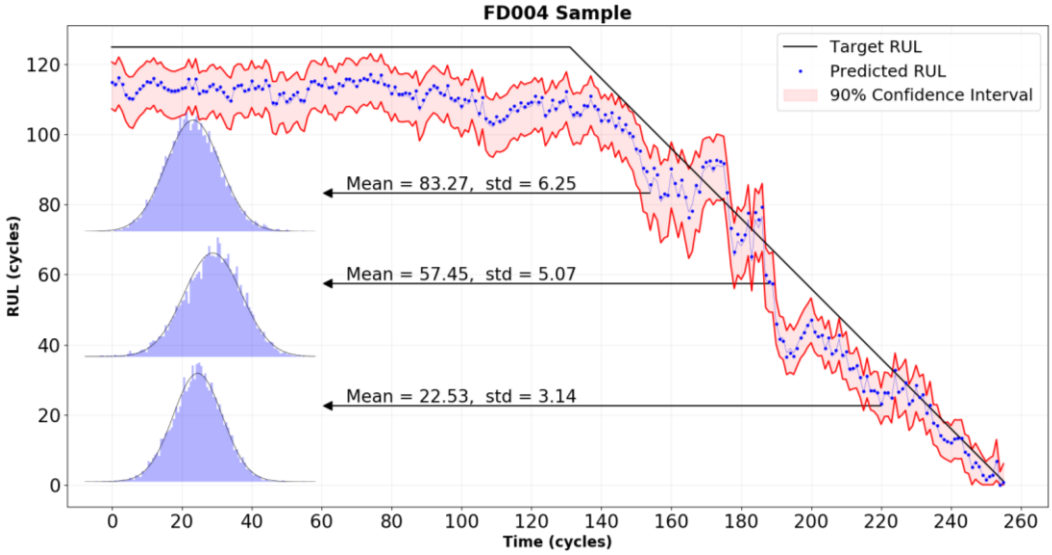

Figure 5.6: RUL estimation under uncertainty. Engine sample from FD004.

Chapter 6

CS2 Lithium-Ion Batteries

As it was done for the C-MAPSS dataset, a grid search is performed using the data from the CS2 lithium-ion battery cells to select the most important hyperparameters of the proposed CNNBiLSTM architecture, as described in Chapter 4. In this case, different models are trained for all possible combinations of the hyperparameters listed in Table 4.3, choosing those which yield the lowest SOC and SOH simultaneously. The selected hyperparameters are presented in Table 6.1. Once again, the grid search shows that the training time is directly proportional to the number of hidden units in the LSTM cells, as well as to the batch size.

Table 6.1: Selected architecture's hyperparameters for the C-MAPSS dataset.

Feature Maps 1st Convolution	Feature Maps 2nd Convolution	Epochs	Batch Size	Number Hidden Units LSTM	Dropout Probability
64	128	20	256	50	0.6

6.1 Model Training, Performance and Comparison

Five models are trained applying a 40% dropout after each layer as described in Chapter 4. One model is trained for each battery cell individually and a fifth model is trained using all four battery cells simultaneously. The data used to train and validate the model are randomly selected, and the data for testing is not used in any way during the training procedure. To avoid overfitting, models are trained until either the maximum number of epochs is reached, or the validation error does not improve for three consecutive epochs. Each model is trained 10 different times to obtain a mean and standard deviation from the training procedure.

The RMSE results for the SOC and SOH of each model are provided in Table 6.2. It can be seen that for each model, the standard deviation of the RMSE is small compared to its average value. Thus, the proposed architecture shows robustness on the training procedure. Table 6.2 also shows that models can accurately estimate both SOC and SOH, where the highest error is associated with SOH estimates for Model 2, which is trained with data from cell #36 (3.33% average RMSE). It is important to note that Model 2 also yields the highest average RMSE in SOC estimates, which is only in the order of 1.98%. When using all four cells in the training stage, Model 5 successfully generalizes the degradation of the batteries as well as the estimation of remaining energy during

the discharge cycle, as it can be corroborated by observing that the average error is lower than three of the other models for both SOC and SOH.

Table 6.2: CNNBiLSTM model training RMSE results for the SOC and SOH.

		Proposed CNNBiLSTM Model			
		SOC		SOH	
Dataset	**Model #**	**Mean**	**STD**	**Mean**	**STD**
CS2 35	Model 1	1.19	0.25	2.01	0.38
CS2 36	Model 2	1.98	0.51	3.33	0.63
CS2 37	Model 3	1.69	0.27	2.70	0.56
CS2 38	Model 4	1.71	0.28	2.96	0.73
All cells	Model 5	1.58	0.14	2.49	0.36

Other research efforts mainly study the CS2 lithium-ion battery dataset for the RUL estimation, either through conventional methods [88] or deep learning methods such as LSTM [100]. However, only one regression model has been developed for the joint estimation of the SOC and SOH. Indeed, Huang et al. [102] proposes a polynomial regression method for an online estimation model of both SOC and SOH, where the estimation of the SOH is directly dependent on the previous estimation of the SOC. This dependency is undesired because of the cumulative error associated to each regression model as well as the possible bias of the adjusted parameters to the selected training data. Furthermore, in the approach put forward in [102], models were trained using all data from each battery cell and tested with samples taken every 25 discharge cycles, starting from N=25 cycles up to N=700 cycles, which is said to be the most relevant interval with respect to the functionality of the battery, and where the variables' behavior is more stable. It is important to note that this chosen evaluation leads to biased results since the resulting model is tested with data used in the training process. Moreover, the arbitrary selection of a determined range to test the models guarantees lower RMSE results.

In contrast, the proposed CNNBiLSTM model is trained, validated and tested with independent randomly selected data from each battery cell. Therefore, in order to compare the performance of the proposed deep learning framework with [102], the RMSE metric is calculated for each trained model based on randomly chosen test sets for each battery cell. These results are given in Table 6.3. It can be observed that when compared with [102], the proposed CNNBiLSTM-based model outputs similar results for the RMSE when evaluating each model with its corresponding test set (i.e., Table 6.3 diagonal). However, in [102] the test sets consider data within the train set, in an operational range with more homogeneous data and less uncertainty. Table 6.3 also shows the results yielded by Model 5, which is trained using training data from all battery cells and delivers results that outperforms all model evaluations from the regression model. That is, Model 5 is tested with more heterogeneous data than the other models, since the data is randomly taken from all four battery cells, and yet delivers better performance results than any other model.

Hence, the proposed CNNBiLSTM has a good generalization capability when dealing with non-homogeneous data and with high variability. Indeed, Model 5 outputs a RMSE equal to 1.40% when evaluated with all test sets, which is smaller than any of the regression models evaluated in the different datasets. Also, given the smaller standard deviation obtained when training the CNNBiLSTM model with data from all battery cells, it can be argued that the proposed model is robust at training, and has the capability to obtain a better generalization when trained for a bank of batteries.

Table 6.3: RMSE [%] SOC comparison when evaluating proposed CNNBiLSTM model with test set.

	Tested Cell #	**35**	**36**	**37**	**38**	**All**
	Method	**RMSE**	**RMSE**	**RMSE**	**RMSE**	**RMSE**
Model 1	Hung	1.50	2.40	1.90	1.40	-
	CNNBiLSTM	1.56	1.63	1.73	2.50	1.91
Model 2	Hung	2.00	1.90	2.10	1.90	-
	CNNBiLSTM	1.96	1.79	1.95	3.36	2.38
Model 3	Hung	1.60	2.20	1.80	1.50	-
	CNNBiLSTM	2.25	2.10	1.63	2.99	2.31
Model 4	Hung	1.60	2.40	1.90	1.40	-
	CNNBiLSTM	2.34	2.39	2.10	1.41	2.08
Model 5	Hung	-	-	-	-	-
	CNNBiLSTM	1.47	1.4	1.36	1.36	1.40

6.2 Uncertainty Measurement on the Models' Estimation

Uncertainty characterization is a key element in the state estimation problem. Particularly, it is important to validate assumptions on the structure and probability distribution associated with the model error. For the CS2 battery cells, the main assumption is that modelling errors can be characterized as additive Gaussian noise; in other words, that the RMSE is an adequate metric for uncertainty characterization of SOC and SOH estimates. Hence, two different approaches are applied to validate this assumption, as well as to quantify the precision of SOC and SOH estimates in the proposed CNNBiLSTM-based models.

On the one hand, the test set from each cell is evaluated 5,000 times through their respective trained model using a 10% dropout probability. Then, the RMSE results obtained for the SOC and SOH of each trained model are used to fit a Gaussian distribution. In it, its standard deviation reflects the uncertainty of each model on its estimated values. Table 6.4 summarizes the results for this approach. Furthermore, Figure 6.1 and Figure 6.2 show the obtained distribution for the SOC and SOH when evaluating Models 1 to 4, and Figure 6.3 shows the obtained distribution for the SOC and SOH when evaluating Model 5 (trained for all datasets).

The Shapiro-Wilk null hypothesis test [132] is performed to evaluate the normality of the yield distributions. The null hypothesis states that the data was given by a Gaussian distribution, where this hypothesis is rejected for a low test-statistic (*W*) value, or if the *p-value* is lower than the desired significance level. As it can be seen from Table 6.4, all *W* values yield by the distributions are 1.0 (the best possible result) with a relative high *p-value*. Hence, the null hypothesis is accepted, and it can be stated that the uncertainty over the RMSE when applying dropout in each model yields a Gaussian distribution.

Table 6.4: Uncertainty measures in the CS2 test sets through dropout.

		SOC				**SOH**			
Cell #	**Model #**	**Mean**	**STD**	**W**	**p-value**	**Mean**	**STD**	**W**	**p-value**
35	Model 1	2,37	0,02	1,00	0,65	3,51	0,03	1,00	0,50
36	Model 2	2,70	0,02	1,00	0,95	4,48	0,02	1,00	0,69
37	Model 3	2,61	0,02	1,00	0,49	3,57	0,02	1,00	0,71
38	Model 4	2,67	0,02	1,00	0,26	3,97	0,02	1,00	0,37
All	Model 5	2,45	0,01	1,00	0,71	3,78	0,01	1,00	0,64

From Table 6.4, it can also be observed that the obtained mean RMSE for each dataset is higher than the ones reported in Table 6.3, although with smaller standard deviation. In other words, although model errors are slightly affected, precision on the resulting predicted values is high. This error increase could be associated to the few number of features that the model receives as input

during the dropout-based uncertainty quantification procedure, since the action of "turning off" neurons leads to information loss. Note that the Gaussian distribution obtained for the evaluation of each test set is a direct result from applying dropout while testing, and it was not imposed in any way to the models presented in Table 6.4 during the training procedure.

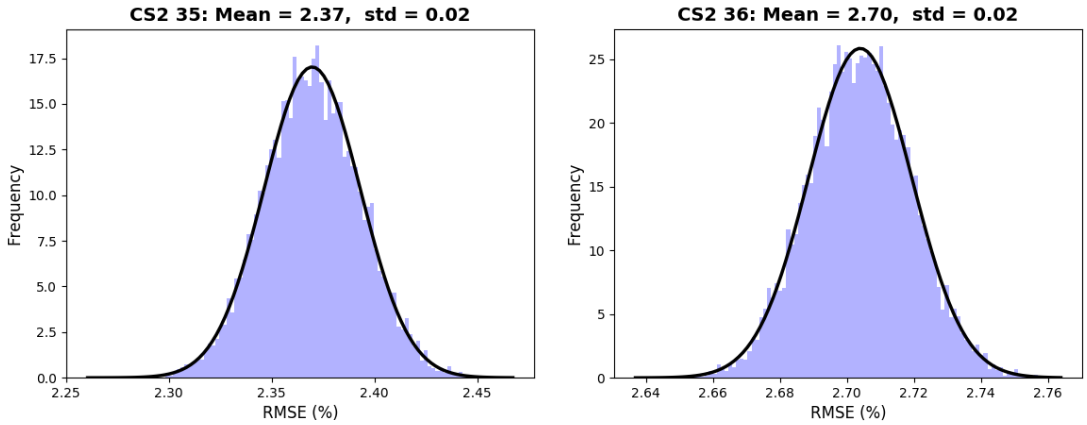

Figure 6.1: RMSE uncertainty for cells 35 and 36.

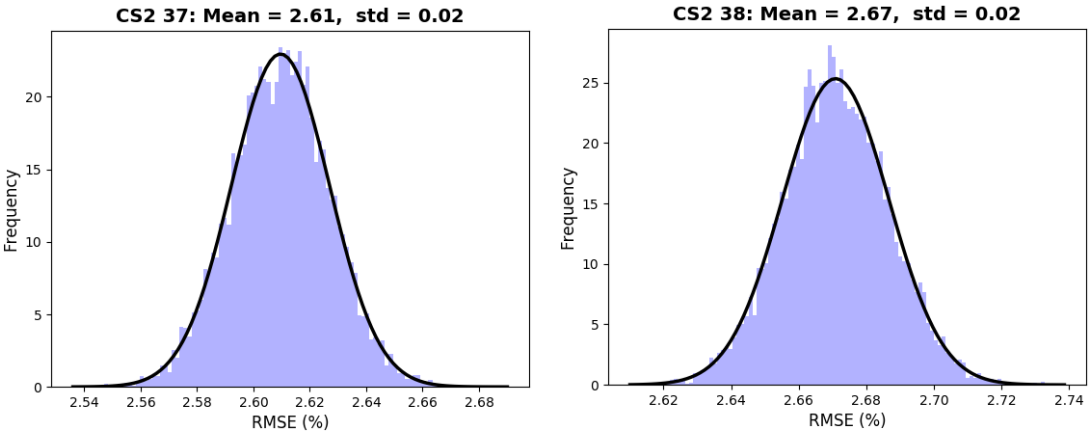

Figure 6.2: RMSE uncertainty for cells 37 and 38.

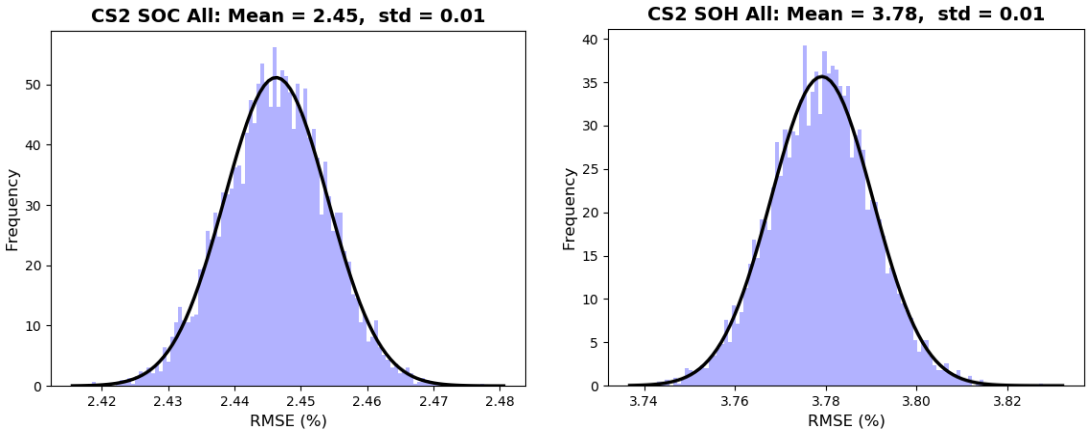

Figure 6.3: SOC and SOH test RMSE uncertainty for Model 5, trained for all battery cells.

On the other hand, it is also important to evaluate the uncertainty associated with SOC estimates when using the trained models for real-time battery assessment. For this purpose, an uncertainty quantification procedure is also implemented based on a 10% dropout probability and using random discharge cycles from each battery cell. In these cases, the 2D matrices that characterize each discharge cycle are generated as described in Chapter 4. Each 2D matrix sample from each discharge cycle is evaluated 10,000 times in through its corresponding trained model. Figure 6.4 to Figure 6.6 show the estimated SOC at each point of the discharge cycle for each sample. The estimation SOC is considered as the average of the 10,000 SOC yielded by the model for each 2D matrix sample. Furthermore, a 90% probability interval is given for each estimation point.

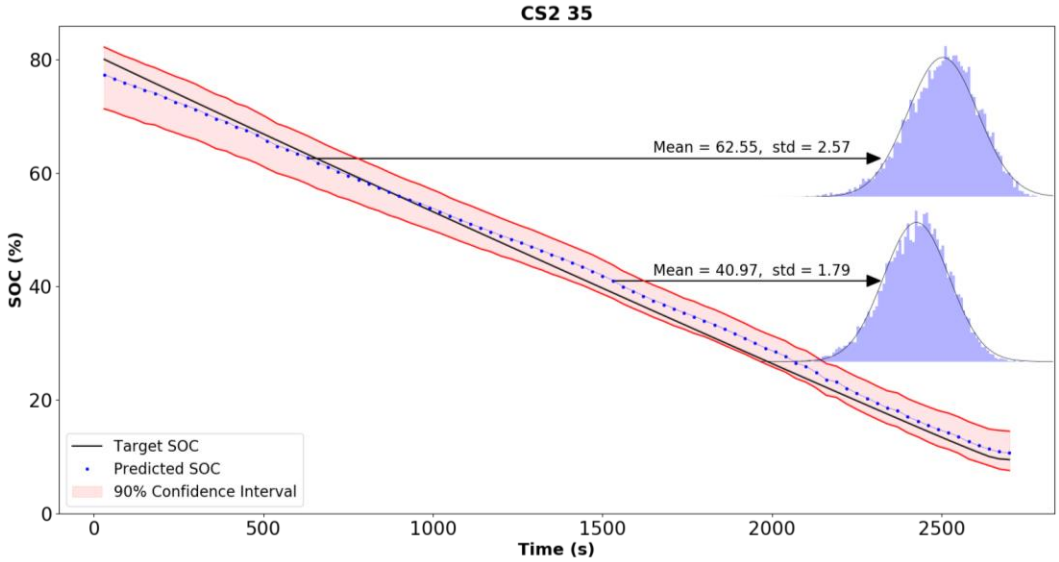

Figure 6.4: SOC estimation with uncertainty from Model 1. Discharge sample from cell 35.

From this approach, it can be seen that each estimated SOC point represents the mean value of a Gaussian distribution, such as the ones obtained in Figure 6.1 and Figure 6.2, where its standard deviation represents the uncertainty over the estimation of the SOC. It can also be observed that the closer the estimation is to the complete discharged event, the narrower is the 90% probability interval. Thus, the estimation precision increases when approaching a fully discharged battery. This is due to the time-window length to create the 2D sample matrices, given that there are more samples representing a 70% to 100% discharged battery than to a fully charged battery.

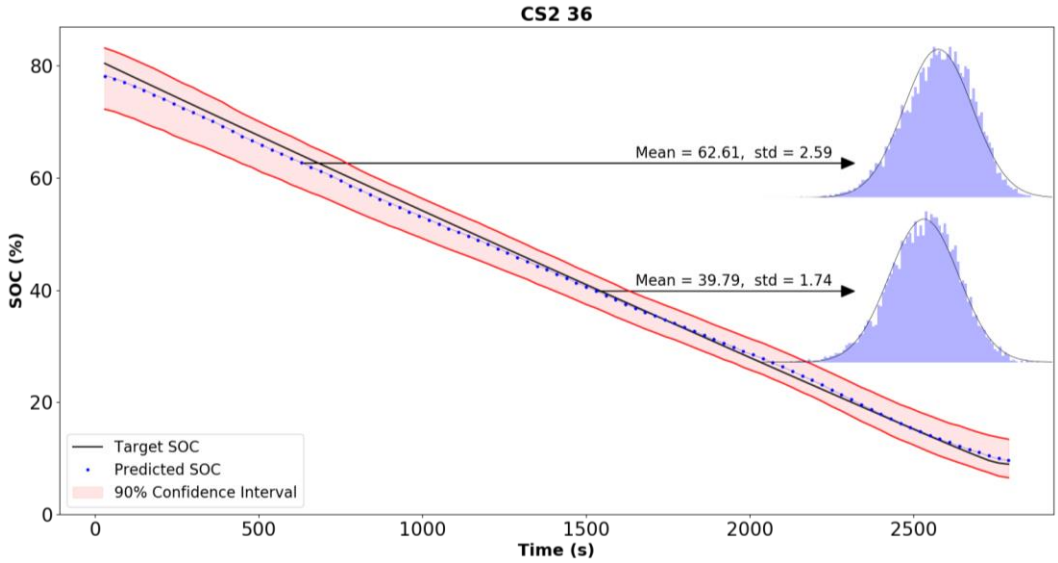

Figure 6.5: SOC estimation with uncertainty from Model 2. Discharge sample from cell 36.

The measured uncertainties can be related to many sources implicitly involved on the SOC and SOH estimation. For instance, the precision on the measured variables, especially those used to train and evaluate the model such as the voltage and current. Measurement devices often come with an associated error, which can be further contaminated with noise coming from the environment, as well as from measuring different battery cells with different measurement instruments. Another possible source of uncertainty for the estimated results can be related to the low sampling frequency of the studied dataset. Indeed, each 2D sample matrix evaluated through the proposed CNNBiLSTM model contains 24 time-steps, equivalent to 240s. Hence, uncertainty on the estimated SOC and SOH may be reduced if a higher sampling frequency is used to measure the variables.

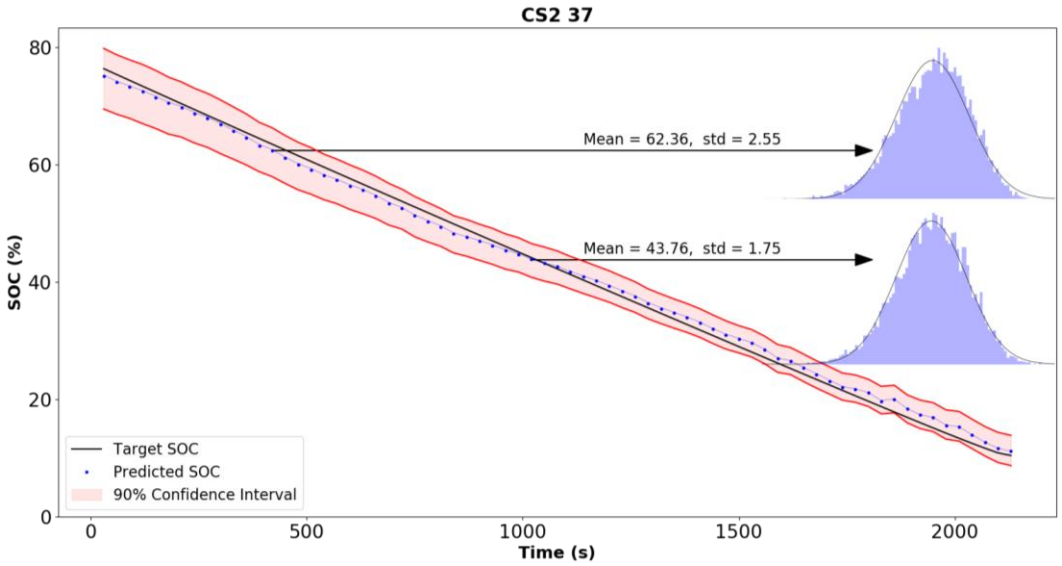

Figure 6.6: SOC estimation with uncertainty from Model 3. Discharge sample from cell 37.

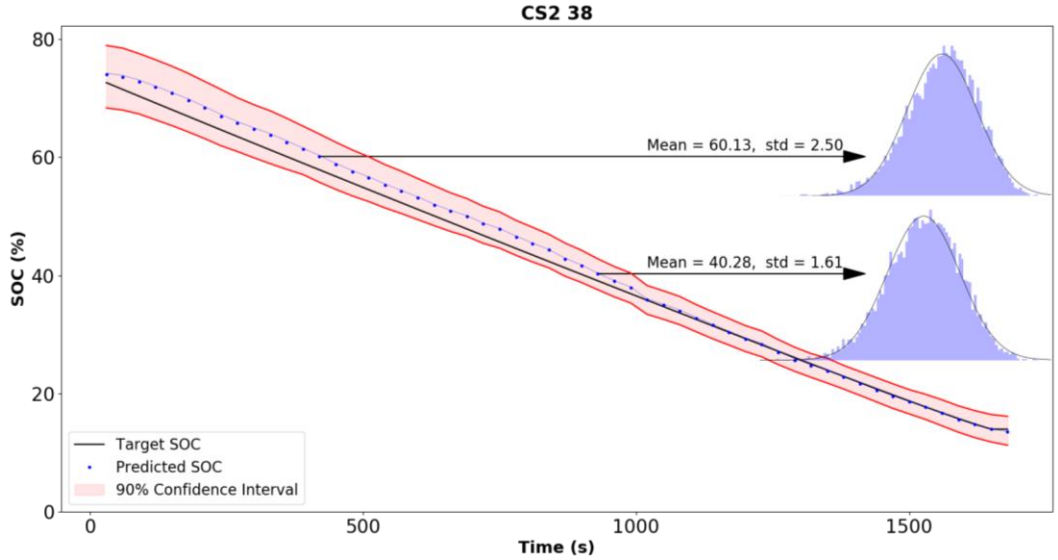

Figure 6.7: SOC estimation with uncertainty fromModel 4. Discharge sample from cell 38.

Chapter 7

Concluding Remarks

In this thesis, a deep learning framework is presented for the health state estimation of complex systems based on big machinery data. The framework is validated through two different datasets and consists of two CNN layers aiming to obtain an abstract relationship between sensor measurements, combined with a bidirectional LSTM layer capable of understanding the temporal behavior of such sensors. Thus, the proposed deep learning framework is named CNNBiLSTM. The C-MAPSS Turbofan Engines and the CS2 Lithium-Ion Battery Cells datasets are used to validate the proposed architecture through the training and testing of different models. Furthermore, an estimation of the uncertainty for the models' prediction is given through dropout as a Bayesian approximation. Noteworthy results are yielded by the proposed framework, which are addressed in the following conclusions and comments.

7.1 Conclusions

During the last decade, deep learning methods have become popular in engineering applications, particularly in data analysis for reliability assessment, which used to be an inefficient process due to the need of expert knowledge on the studied system, along with the limitation of traditional PHM techniques. Reliability related, data-driven applications still have many challenges to confront in order to keep improving the estimation of health state parameters which can guarantee an accurate diagnosis for systems, equipment and, more importantly, the safety of the industrial working staff.

In the context of estimating and predicting the state of health of industrial machinery, a deep learning framework is proposed, aiming to analyze sensorial data and obtain a temporal relationship within measured variables. From a data analysis point of view, the approach combines the capability of Convolutional and Recurrent Neural Networks to analyze spatial and temporal data, respectively. Moreover, the architecture of the proposed CNNBiLSTM framework is also thought from an engineering point of view. The convolutional filters from both CNN layers are applied to independently process the sensorial data at each time-step. The stride chosen for the CNN filters process a 2D matrix sample of multisensorial data in time to give an abstract representation of the relationship among all sensors. Thus, for each time-step, sensor measurements are paired in couples through the first convolution, to later obtain a relationship among all sensors through the second convolutional filter. Analyzing 2D matrices comes as a strong asset from a reliability point of view, given that the trained models do not need any information from the previous behavior or state of

the studied system. This allows trained models to yield an accurate health state assessment of the system with only a few datapoints.

The proposed framework shows robustness for both prognosis of the remaining useful life for rotational machinery, as well as for the joint estimation of the State of Charge and State of Health of lithium-ion batteries, based on multivariate sensor data. For the C-MAPSS dataset, one model is trained and tested for each sub-dataset. The average results obtained for both metrics, RMSE and Score, were compared with other state of the art architectures, where the proposed CNNBiLSTM-based models showed an improvement on the RMSE and similar Score values over the test sets. Moreover, when comparing the results obtained (Table 5.2) for sub-datasets FD002 and FD004 with sub-datasets FD001 and FD003, it becomes clear that the proposed architecture is more sensitive to operational conditions rather than the number of fault modes. It is to be noticed that unlike other studies, the test set was not modified for the testing procedure to meet the R_{early} criteria. Hence, results for both RMSE and Score could be further improved without modifying the proposed architecture. However, this approach is not correct, since test sets should never be modified when testing data analysis models.

Uncertainty on the obtained results was estimated through two approaches. For the first approach, uncertainty over the RMSE results is estimated by applying a 10% dropout on 5,000 forward passes of the test sets through the trained network. Results presented in Table 5.3 along with Figure 5.1 and Figure 5.2, show a smooth Gaussian distribution with a small standard deviation for all four models (Table 5.3). Thus, the proposed CNNBiLSTM-based models are robust on their accuracy to predict the RUL of a turbofan engine. For the second approach, a visualization is presented for the RUL estimation with its uncertainty for an entire run from one sample engine from each sub-dataset. Figure 5.3 to Figure 5.6 report a small uncertainty on the models' predictions, which decreases towards the failure event. These figures also show that for those engines where only one operational condition was reported, the proposed model tends to be conservative, given that the predicted values for the RUL were located under the target curve. Thus, the small uncertainty on the models' results makes the proposed CNNBiLSTM model both accurate and precise on its RUL predictions.

To analyze the CS2 lithium-ion batteries from CALCE-UMD, five different models are trained from the dataset: one for each battery cell and a fifth using all cells simultaneously. The proposed CNNBiLSTM-based models show an excellent performance after training, where all models obtain an average RMSE under 2.0% for the SOC and below 3.5% for the SOH. These results are compared with a parametric regression model which gives a SOH directly dependent from the SOC. Here, Model 5, trained and tested for all battery cells, outperforms the parametric regression model (Table 6.3). Furthermore, the CNNBiLSTM-based model is tested with data randomly selected from each dataset, whereas the parametric regression model not only uses the same data to train and test the model, but it also takes only data from a certain range of cycle degradation where the batteries behavior is more homogeneous. Hence, it is fair to conclude that the proposed framework stands as a better approach.

Two uncertainty measurements are given for the SOC estimation by applying several forward passes through the model with a 10% dropout probability. A small standard deviation is obtained on the tests RMSE (Table 6.4), showing a low estimation uncertainty for the proposed CNNBiLSTM models. Moreover, a second approach to visualize the uncertainty of the model is given by predicting an entire discharge cycle for each battery cells, where a 90% confidence interval is shown with a small standard deviation (Figure 6.4 to Figure 6.7). It can be concluded that the uncertainty of the model is small for all predictions, and it diminishes towards the complete discharge event.

The proposed framework presents several advantages when compared with traditional methods for the joint estimation of the SOC and SOH in battery cells. First, only a small time-window of 24 time-steps is needed to yield an accurate estimation of the state of the battery cell, making the model suitable for online estimation. This is a remarkable result, also given the fact that the studied CS2 dataset has a low data sample frequency of 0.1 Hz. Furthermore, the trained models do not require the number of charge-discharge cycles that the battery has performed before. Therefore, the SOC and SOH estimation come only from the relationship of the measured variables. Lastly, the SOH estimation is completely independent from the SOC estimation, thus results are less prone to be biased and to have cumulative error.

From the presented results for both datasets, C-MAPPS and CS2, it can be concluded that the trained models successfully combine the spatial recognition capabilities of CNN with the sequential data processing of bidirectional LSTM. This is reflected in the good results yielded for the RMSE and Score metrics when testing the C-MAPSS turbofan engines, as well as the RMSE for the CS2 lithium-ion battery cells dataset. These results obtained for the test sets are a compelling proof that the proposed CNNBiLSTM framework can be a powerful asset for reliability purposes in big datasets composed by multisensorial measurements. This conclusion is reinforced when considering the time required to train and test these models. Indeed, training one model for the C-MAPSS dataset takes an average of 64.3s, while for the CS2 dataset the training time averages 54.4s. Testing a new 2D sample matrix takes an average of 0.01s for both datasets, which advocates that the trained models could be used for online estimation if a time-window measurement is available.

Finally, the proposed CNNBiLSTM trained models show improvement from the state-of-the-art results reported to date for both datasets studied. Even though the improvement over these results might not be significant enough, it must also be considered that the proposed framework does not need any expert knowledge on the studied systems to give an accurate assessment. This comes as a more notable result when considering that the proposed CNNBiLSTM framework was used to train two completely different datasets, which present different behaviors and operate under diverse conditions, giving accurate results for both challenges. Hence, it is clear that the proposed CNNBiLSTM framework can be used for reliability assessment of any monitored equipment.

7.2 Future Work

The results presented and discussed above are a compelling proof of the framework's feature extraction and sequential analysis capabilities from raw data. The framework can be adapted to different circumstances and yield accurate results. However, there are still some aspects where improvement can be made. For instance, one of the main drawbacks from the proposed CNNBiLSTM framework is that the training procedure is fully supervised. Hence, although models can be directly applied to many challenges, there are still many reliability related phenomena where it is not possible to generate the necessary amount of labeled data to train a fully supervised model. Thus, a next step for the presented work would be to implement an unsupervised data preprocessing technique able to generate the labels for the training data, i.e., by applying traditional ML techniques such a clustering; or perhaps some more elaborated ones such as Variational Autoencoders or Generative Adversarial Networks.

Another interesting follow-up to the presented CNNBiLSTM framework is to implement an evaluation software which could be used to continuously monitor the state of health of a system through data acquired online. This can be accomplished by developing a software with a friendly Graphical User Interface (GUI), aiming to yield real-time results, as well as updating the trained models with the new incoming data.

Bibliography

[1] L. Yao and Z. Ge, "Deep Learning of Semisupervised Process Data With Hierarchical Extreme Learning Machine and Soft Sensor Application," vol. 65, no. 2, pp. 1490–1498, 2018.

[2] O. Costilla-reyes, P. Scully, and K. B. Ozanyan, "Deep Neural Networks for Learning Spatio-Temporal Features From Tomography Sensors," vol. 65, no. 1, pp. 645–653, 2018.

[3] J. Pan, Y. Zi, J. Chen, Z. Zhou, and B. Wang, "LiftingNet: A Novel Deep Learning Network with Layerwise Feature Learning from Noisy Mechanical Data for Fault Classification," *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 4973–4982, 2017.

[4] D. Lund, C. MacGillivray, V. T.-… (IDC), T. Rep, and undefined 2014, "Worldwide and regional internet of things (iot) 2014–2020 forecast: A virtuous circle of proven value and demand," *pdfs.semanticscholar.org*.

[5] Y. Lei, F. Jia, J. Lin, … S. X.-I. T. on, and undefined 2016, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *ieeexplore.ieee.org*.

[6] Z. Hu and P. Jiang, "An Imbalance Modified Deep Neural Network with Dynamical Incremental Learning for Chemical Fault Diagnosis," *IEEE Trans. Ind. Electron.*, vol. 0046, no. c, pp. 1–1, 2018.

[7] G. Zhao, G. Zhang, Y. Liu, B. Z.-… (ICPHM), 2017 IEEE, and undefined 2017, "Lithium-ion battery remaining useful life prediction with Deep Belief Network and Relevance Vector Machine," *ieeexplore.ieee.org*.

[8] M. Ma, C. Sun, X. C.-I. T. on Industrial, and undefined 2018, "Deep Coupling Autoencoder for Fault Diagnosis with Multimodal Sensory Data," *ieeexplore.ieee.org*.

[9] Z. Chen, C. Li, and R.-V. Sanchez, "Gearbox Fault Identification and Classification with Convolutional Neural Networks," *Shock Vib.*, vol. 2015, pp. 1–10, 2015.

[10] M. Zhao and M. Kang, "Deep Residual NetworksWith Dynamically WeightedWavelet Coefficients for Fault Diagnosis of Planetary Gearboxes," vol. 65, no. 5, pp. 4290–4300, 2018.

[11] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mech. Syst. Signal Process.*, vol. 72–73, pp. 303–315, 2016.

[12] W. Yan, D. Tang, and Y. Lin, "IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS A Data-Driven Soft Sensor Modeling Method Based on Deep Learning and its Application," vol. 0046, no. c, pp. 4237–4245, 2016.

[13] Z. Chen, R. Zhao, Q. Zhu, M. K. Masood, Y. C. Soh, and K. Mao, "Building Occupancy Estimation with Environmental Sensors via CDBLSTM," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9549–9559, 2017.

[14] O. Janssens *et al.*, "Convolutional Neural Network Based Fault Detection for Rotating Machinery," *J. Sound Vib.*, vol. 377, pp. 331–345, 2016.

[15] H. Oh, J. H. Jung, B. C. Jeon, and B. D. Youn, "Scalable and Unsupervised Feature Engineering Using Vibration-Imaging and Deep Learning for Rotor System Diagnosis," *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 3539–3549, 2017.

[16] H. Shao, H. Jiang, H. Zhang, and T. Liang, "Electric Locomotive Bearing Fault Diagnosis

Using a Novel Convolutional Deep Belief Network," vol. 65, no. 3, pp. 2727–2736, 2018.
[17] D. Verstraete *et al.*, "Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings," *Shock Vib.*, vol. 2017, pp. 1–17, Oct. 2017.
[18] C. Lin, H. Mu, R. Xiong, and J. Cao, "Multi-model probabilities based state fusion estimation method of lithium-ion battery for electric vehicles: State-of-energy," *Appl. Energy*, vol. 194, pp. 560–568, 2017.
[19] W. He, N. Williard, C. Chen, and M. Pecht, "State of charge estimation for electric vehicle batteries using unscented kalman filtering," *Microelectron. Reliab.*, vol. 53, no. 6, pp. 840–847, 2013.
[20] J. Yan, G. Xu, H. Qian, and Y. Xu, "Robust state of charge estimation for hybrid electric vehicles: Framework and algorithms," *Energies*, vol. 3, no. 10, pp. 1654–1672, 2010.
[21] H. He, X. Zhang, R. Xiong, Y. Xu, and H. Guo, "Online model-based estimation of state-of-charge and open-circuit voltage of lithium-ion batteries in electric vehicles," *Energy*, vol. 39, no. 1, pp. 310–318, 2012.
[22] R. Yang, R. Xiong, H. He, H. Mu, and C. Wang, "A novel method on estimating the degradation and state of charge of lithium-ion batteries used for electrical vehicles," *Appl. Energy*, vol. 207, pp. 336–345, 2017.
[23] Y. Zou, X. Hu, H. Ma, and S. E. Li, "Combined State of Charge and State of Health estimation over lithium-ion battery cell cycle lifespan for electric vehicles," *J. Power Sources*, vol. 273, pp. 793–803, 2015.
[24] Y. Zheng, M. Ouyang, X. Han, L. Lu, and J. Li, "Investigating the error sources of the online state of charge estimation methods for lithium-ion batteries in electric vehicles," *J. Power Sources*, vol. 377, no. December, pp. 161–188, 2018.
[25] R. Xiong, L. Li, Z. Li, Q. Yu, and H. Mu, "An electrochemical model based degradation state identification method of Lithium-ion battery for all-climate electric vehicles application," *Appl. Energy*, vol. 219, no. 5, pp. 264–275, 2018.
[26] P. Espinoza, A. Pérez, M. Orchard, H. Navarrete, and D. Pola, "A simulation engine for predicting state-of-charge and state-of-health in lithium-ion battery packs of electric vehicles," *Annu. Conf. Progn. Heal. Manag. Soc. 2017*, p. 16, 2017.
[27] H. Sheng and J. Xiao, "Electric vehicle state of charge estimation: Nonlinear correlation and fuzzy support vector machine," *J. Power Sources*, vol. 281, pp. 131–137, 2015.
[28] M. A. Hannan, M. S. H. Lipu, A. Hussain, and A. Mohamed, "A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations," *Renew. Sustain. Energy Rev.*, vol. 78, no. May, pp. 834–854, 2017.
[29] W. S. Putra, B. R. Dewangga, A. Cahyadi, and O. Wahyunggoro, "Current estimation using Thevenin battery model," in *Proceedings - Joint International Conference on Electric Vehicular Technology and Industrial, Mechanical, Electrical and Chemical Engineering, ICEVT 2015 and IMECE 2015*, 2016, pp. 5–9.
[30] L. W. Kang, X. Zhao, and J. Ma, "A new neural network model for the state-of-charge estimation in the battery degradation process," *Appl. Energy*, vol. 121, pp. 20–27, 2014.
[31] J. Chen, Q. Ouyang, C. Xu, and H. Su, "Neural Network-Based State of Charge Observer Design for Lithium-Ion Batteries," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 1, pp. 313–320, 2018.
[32] L. Xu, J. Wang, and Q. Chen, "Kalman filtering state of charge estimation for battery management system based on a stochastic fuzzy neural network battery model," *Energy Convers. Manag.*, vol. 53, no. 1, pp. 33–39, 2012.
[33] I. H. Li, W. Y. Wang, S. F. Su, and Y. S. Lee, "A merged fuzzy neural network and its

applications in battery state-of-charge estimation," *IEEE Trans. Energy Convers.*, vol. 22, no. 3, pp. 697–708, 2007.
[34] M. A. Hannan, M. S. H. Lipu, A. Hussain, M. H. Saad, and A. Ayob, "Neural Network Approach for Estimating State of Charge of Lithium-ion Battery Using Backtracking Search Algorithm," *IEEE Access*, vol. 6, pp. 1–1, 2018.
[35] W. He, N. Williard, C. Chen, and M. Pecht, "State of charge estimation for Li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation," *Int. J. Electr. Power Energy Syst.*, vol. 62, pp. 783–791, 2014.
[36] M. Charkhgard and M. Farrokhi, "State-of-charge estimation for lithium-ion batteries using neural networks and EKF," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4178–4187, 2010.
[37] Z. Chen, S. Qiu, M. A. Masrur, and Y. L. Murphey, "Battery state of charge estimation based on a combined model of Extended Kalman Filter and neural networks," *2011 Int. Jt. Conf. Neural Networks*, pp. 2156–2163, 2011.
[38] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. 1991.
[39] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, First Edit. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2017.
[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE, 86(11)2278-2324*, 1998.
[41] Y. LeCun, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[42] A. YoshuaBengio, Ian J.Goodfellow, *Deep Learning*. 2015.
[43] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," pp. 1–19, 2017.
[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[45] J. Schmidhuber, "Deep learning in Neural networks: An overview," *Neural Networks*, vol. 61, no. 1. pp. 85–117, 2015.
[46] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1–32, 1997.
[47] W. Li, G. Wu, S. Member, and F. Zhang, "Hyperspectral Image Classification Using Deep Pixel-Pair Features," *ieeexplore.ieee.org*, pp. 1–10, 2016.
[48] C. Szegedy *et al.*, "Going Deeper with Convolutions," pp. 1–9, 2014.
[49] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using Convolutional Networks," *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 648–656, 2015.
[50] C. Zhang *et al.*, "A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017.
[51] S. Ren, K. He, R. Girshick, J. S.-I. transactions on pattern, and undefined 2017, "Faster R-CNN: towards real-time object detection with region proposal networks," *ieeexplore.ieee.org*.
[52] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017, pp. 131–135.
[53] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9642, pp. 214–

228, 2016.
[54] L. Wen, X. Li, L. Gao, and Y. Zhang, "A New Convolutional Neural Network Based Data-Driven Fault Diagnosis Method," *IEEE Trans. Ind. Electron.*, vol. 0046, no. c, 2017.
[55] D. Lee, V. Siu, R. Cruz, and C. Yetman, "Convolutional Neural Net and Bearing Fault Analysis," *Int'l Conf. Data Min. | DMIN'16 |*, pp. 194–200, 2016.
[56] F.-C. Chen and R. M. R. Jahanshahi, "NB-CNN: Deep Learning-based Crack Detection Using Convolutional Neural Network and Naïve Bayes Data Fusion," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 1–1, 2017.
[57] D. Suryani, P. Doetsch, and H. Ney, "On the benefits of convolutional neural network combinations in offline handwriting recognition," *Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR*, pp. 193–198, 2017.
[58] W. J. Murdoch and A. Szlam, "Automatic Rule Extraction from Long Short Term Memory Networks," no. 2016, pp. 1–12, 2017.
[59] N. Laptev, J. Yosinski, L. Erran, and L. Slawek, "Time-series Extreme Event Forecasting with Neural Networks at Uber," *Time Ser. Work. @ ICML 2017*, 2017.
[60] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 2014.
[61] J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder–decoder," *Reliab. Eng. Syst. Saf.*, vol. 170, no. March 2017, pp. 73–82, 2017.
[62] Z. Zhao *et al.*, "Multi-bearing remaining useful life collaborative prediction: A deep learning approach," *J. Manuf. Syst.*, vol. 43, no. 1, pp. 248–256, 2017.
[63] R. Zhao, J. Wang, R. Yan, and K. Mao, "Machine health monitoring with LSTM networks," *Proc. Int. Conf. Sens. Technol. ICST*, 2016.
[64] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional Bi-directional LSTM networks," *Sensors (Switzerland)*, vol. 17, no. 2, pp. 1–19, 2017.
[65] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation," *2017 IEEE Int. Conf. Progn. Heal. Manag.*, pp. 88–95, 2017.
[66] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 0, pp. 1–13, 2017.
[67] M. Yuan, Y. Wu, and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network," *Int. Conf. Aircr. Util. Syst.*, pp. 135–140, 2016.
[68] H. Liu, L. Li, and J. Ma, "Rolling Bearing Fault Diagnosis Based on STFT-Deep Learning and Sound Signals," *Shock Vib.*, vol. 2016, 2016.
[69] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, "Predicting Remaining Useful Life using Time Series Embeddings based on Recurrent Neural Networks," *CEUR Workshop Proc.*, vol. 1828, pp. 89–93, 2017.
[70] P. Malhotra *et al.*, "Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder," 2016.
[71] A. Saxena, M. Ieee, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Prognostics," *Response*, 2008.
[72] X. Li, Q. Ding, and J. Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliab. Eng. Syst. Saf.*, vol. 172, no. October 2017, pp. 1–11, 2018.
[73] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based

Regression Approach for Estimation of Remaining Useful Life," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9642, pp. 214–228, 2016.

[74] M. E. Orchard, P. Hevia-Koch, B. Zhang, and L. Tang, "Risk measures for particle-filtering-based state-of-charge prognosis in lithium-ion batteries," *IEEE Trans. Ind. Electron.*, vol. 60, no. 11, pp. 5260–5269, 2013.

[75] B. Xia, H. Wang, Y. Tian, M. Wang, W. Sun, and Z. Xu, "State of charge estimation of lithium-ion batteries using an Adaptive Cubature Kalman filter," *Energies*, vol. 8, no. 6, pp. 5916–5936, 2015.

[76] B. E. Olivares, M. A. Cerda Muñoz, M. E. Orchard, and J. F. Silva, "Particle-filtering-based prognosis framework for energy storage devices with a statistical characterization of state-of-health regeneration phenomena," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 2, pp. 364–376, 2013.

[77] C. Burgos-Mellado, M. E. Orchard, M. Kazerani, R. Cárdenas, and D. Sáez, "Particle-filtering-based estimation of maximum available power state in Lithium-Ion batteries," *Appl. Energy*, vol. 161, pp. 349–363, 2016.

[78] M. Ye, H. Guo, and B. Cao, "A model-based adaptive state of charge estimator for a lithium-ion battery using an improved adaptive particle filter," *Appl. Energy*, vol. 190, pp. 740–748, 2017.

[79] D. E. Acuña and M. E. Orchard, "Particle-filtering-based failure prognosis via sigma-points: Application to Lithium-Ion battery State-of-Charge monitoring," *Mech. Syst. Signal Process.*, vol. 85, pp. 827–848, 2017.

[80] D. A. Pola *et al.*, "Particle-filtering-based discharge time prognosis for lithium-ion batteries with a statistical characterization of use profiles," *IEEE Trans. Reliab.*, vol. 64, no. 2, pp. 710–720, 2015.

[81] C. P. Ley and M. E. Orchard, "Chi-squared smoothed adaptive particle-filtering based prognosis," *Mech. Syst. Signal Process.*, vol. 82, pp. 148–165, 2017.

[82] X. Zhang, Y. Wang, J. Wu, and Z. Chen, "A novel method for lithium-ion battery state of energy and state of power estimation based on multi-time-scale filter," *Appl. Energy*, vol. 216, no. February, pp. 442–451, 2018.

[83] H. Mu, R. Xiong, H. Zheng, Y. Chang, and Z. Chen, "A novel fractional order model based state-of-charge estimation method for lithium-ion battery," *Appl. Energy*, vol. 207, pp. 384–393, 2017.

[84] J. Yu, "Remote Correction Analysis of SOC Accuracy Based On Deep Belief Network," 2017.

[85] J. Yang, B. Xia, W. Huang, Y. Fu, and C. Mi, "Online state-of-health estimation for lithium-ion batteries using constant-voltage charging current analysis," *Appl. Energy*, vol. 212, no. August 2017, pp. 1589–1600, 2018.

[86] J. Li, K. Adewuyi, N. Lotfi, R. G. Landers, and J. Park, "A single particle model with chemical/mechanical degradation physics for lithium ion battery State of Health (SOH) estimation," *Appl. Energy*, vol. 212, no. January, pp. 1178–1190, 2018.

[87] R. Xiong, J. Tian, H. Mu, and C. Wang, "A systematic model-based degradation behavior recognition and health monitoring method for lithium-ion batteries," *Appl. Energy*, vol. 207, pp. 372–383, 2017.

[88] W. He, N. Williard, M. Osterman, and M. Pecht, "Prognostics of lithium-ion batteries based on Dempster-Shafer theory and the Bayesian Monte Carlo method," *J. Power Sources*, vol. 196, no. 23, pp. 10314–10321, 2011.

[89] J. Wu, Y. Wang, X. Zhang, and Z. Chen, "A novel state of health estimation method of Li-

ion battery using group method of data handling," *J. Power Sources*, vol. 327, pp. 457–464, 2016.
[90] B. Saha, K. Goebel, S. Poll, and J. Christophersen, "Prognostics methods for battery health monitoring using a Bayesian framework," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 2, pp. 291–296, 2009.
[91] N. Williard, W. He, M. Osterman, and M. Pecht, "Comparative Analysis of Features for Determining State of Health in Lithium-Ion Batteries," 2013.
[92] S. Li, S. Pischinger, C. He, L. Liang, and M. Stapelbroek, "A comparative study of model-based capacity estimation algorithms in dual estimation frameworks for lithium-ion batteries under an accelerated aging test," *Appl. Energy*, vol. 212, no. January, pp. 1522–1536, 2018.
[93] D. Yang, Y. Wang, R. Pan, R. Chen, and Z. Chen, "State-of-health estimation for the lithium-ion battery based on support vector regression," *Appl. Energy*, no. August, pp. 0–1, 2017.
[94] D. Liu, Y. Luo, Y. Peng, X. Peng, and M. Pecht, "Lithium-ion Battery Remaining Useful Life Estimation Based on Nonlinear AR Model Combined with Degradation Feature," *Annu. Conf. Progn. Heal. Manag. Soc. 2012*, pp. 1–7, 2012.
[95] S. Yin, J. Pang, D. Liu, and Y. Peng, "Remaining Useful Life Prognostics for Lithium-ion Battery Based on Gaussian Processing Regression Combined with the Empirical Model," *Progn. Heal. Manag. Soc.*, 2013.
[96] Y. Liu, G. Zhao, X. Peng, and C. Hu, "Lithium-ion Battery Remaining Useful Life Prediction with Long Short-term Memory Recurrent Neural Network," no. 1, pp. 1–7, 2016.
[97] C. Tampier, A. Pérez, F. Jaramillo, V. Quintero, M. E. Orchard, and J. F. Silva, "Lithium-ion battery end-of-discharge time estimation and prognosis based on Bayesian algorithms and outer feedback correction loops: A comparative analysis," *Proc. Annu. Conf. Progn. Heal. Manag. Soc. PHM*, pp. 182–195, 2015.
[98] Y. Chang, H. Fang, and Y. Zhang, "A new hybrid method for the prediction of the remaining useful life of a lithium-ion battery," *Appl. Energy*, vol. 206, no. May, pp. 1564–1578, 2017.
[99] D. Liu, L. Guo, J. Pang, and Y. Peng, "A Fusion Framework with Nonlinear Degradation Improvement for Remaining Useful Life Estimation of Lithium-ion Batteries," 2011.
[100] Y. Zhang, R. Xiong, H. He, and M. Pecht, "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries," *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, 2018.
[101] T. Kim, W. Qiao, and L. Qu, "Online SOC and SOH estimation for multicell lithium-ion batteries based on an adaptive hybrid battery model and sliding-mode observer," *2013 IEEE Energy Convers. Congr. Expo. ECCE 2013*, pp. 292–298, 2013.
[102] S. C. Huang, K. H. Tseng, J. W. Liang, C. L. Chang, and M. G. Pecht, "An online SOC and SOH estimation model for lithium-ion batteries," *Energies*, vol. 10, no. 4, 2017.
[103] G. L. Plett, "Dual and Joint EKF for Simultaneous SOC and SOH Estimation," *Proc. 21st Electr. Veh. Symp.*, pp. 1–12, 2005.
[104] A. Zenati, P. Desprez, and H. Razik, "Estimation of the SOC and the SOH of Li-ion batteries, by combining impedance measurements with the fuzzy logic inference," in *IECON Proceedings (Industrial Electronics Conference)*, 2010, pp. 1773–1778.
[105] D. Andre, C. Appel, T. Soczka-Guth, and D. U. Sauer, "Advanced mathematical methods of SOC and SOH estimation for lithium-ion batteries," *J. Power Sources*, vol. 224, pp. 20–27, 2013.
[106] "Prediction , estimation & inference of lithium-ion battery state of charge & health simultaneously using machine learning," 2017.
[107] S. Dey, B. Ayalew, and P. Pisu, "Combined estimation of State-of-Charge and State-of-

Health of Li-ion battery cells using SMO on electrochemical model," *Proc. IEEE Work. Appl. Comput. Vis.*, 2014.
[108] K. S. Ng, C. S. Moo, Y. P. Chen, and Y. C. Hsieh, "Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries," *Appl. Energy*, vol. 86, no. 9, pp. 1506–1511, 2009.
[109] G. L. Plett, "Battery management system algorithms for HEV battery state-of-charge and state-of-health estimation," *Adv. Mater. Methods Lithium-Ion Batter.*, vol. 661, no. 2, 2007.
[110] Z. Wei, J. Zhao, D. Ji, and K. J. Tseng, "A multi-timescale estimator for battery state of charge and capacity dual estimation based on an online identified model," *Appl. Energy*, vol. 204, pp. 1264–1274, 2017.
[111] Y. Xing, E. W. M. Ma, K.-L. Tsui, and M. Pecht, "An ensemble model for predicting the remaining useful performance of lithium-ion batteries," *Microelectron. Reliab.*, vol. 53, no. 6, pp. 811–820, 2013.
[112] W. He, N. Williard, M. Osterman, and M. Pecht, "Prognostics of lithium-ion batteries based on Dempster–Shafer theory and the Bayesian Monte Carlo method," *Elsevier*, 2011.
[113] M. H. Rafiei and H. Adeli, "A novel unsupervised deep learning model for global and local health condition assessment of structures," *Eng. Struct.*, 2018.
[114] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Comput. Civ. Infrastruct. Eng.*, 2017.
[115] C. Li, R. V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, "Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis," *Neurocomputing*, vol. 168, pp. 119–127, 2015.
[116] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mech. Syst. Signal Process.*, vol. 104, pp. 799–834, 2018.
[117] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," pp. 214–228, 2016.
[118] L. Ren, J. Cui, Y. Sun, and X. Cheng, "Multi-bearing remaining useful life collaborative prediction: A deep learning approach," *J. Manuf. Syst.*, vol. 43, pp. 248–256, 2017.
[119] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?," no. Nips, 2017.
[120] N. Lotfi, R. G. Landers, S. Member, J. Li, and J. Park, "Reduced-Order Electrochemical Model-Based Uncertainty Estimation," *Ieee Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1217–1230, 2017.
[121] Z. Zhao, Y. Yang, S. Member, S. X. Ding, and L. Li, "Fault-Tolerant Control for Systems With Model Uncertainty and Multiplicative Faults," pp. 1–11, 2017.
[122] E. L. Droguett and A. Mosleh, "Bayesian methodology for model uncertainty using model performance data," *Risk Anal.*, vol. 28, no. 5, pp. 1457–1476, 2008.
[123] E. López Droguett and A. Mosleh, "Bayesian treatment of model uncertainty for partially applicable models," *Risk Anal.*, vol. 34, no. 2, pp. 252–270, 2014.
[124] E. L. Droguett and A. Mosleh, "Integrated treatment of model and parameter uncertainties through a Bayesian approach," *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.*, vol. 227, no. 1, pp. 41–54, 2013.
[125] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," vol. 48, 2015.
[126] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Appendix," pp. 1–20, 2015.
[127] Y. Gal, "Uncertainty in Deep Learning," *PhD Thesis*, no. October, p. 174, 2016.

[128] A. C. Damianou and N. D. Lawrence, “Deep Gaussian Processes,” vol. 31, 2013.
[129] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, “Dropout Improves Recurrent Neural Networks for Handwriting Recognition,” *Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR*, vol. 2014–Decem, no. October, pp. 285–290, 2014.
[130] Y. Gal and Z. Ghahramani, “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks,” 2015.
[131] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, “Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, 2017.
[132] S. S. Shapiro and M. B. Wilk, “An Analysis of Variance Test for Normality,” *Biometrika*, vol. 52, no. 3, pp. 591–6, 1965.